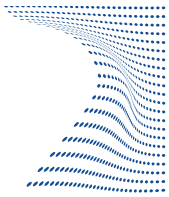
Energy-Accuracy Scaling in Digital ICs: Static and Adaptive Design Methods and Tools

Doctoral Dissertation
Doctoral Program in Computer and Control Engineering (31.st cycle)

Energy-Accuracy Scaling in Digital ICs

Static and Adaptive Design Methods and Tools

Roberto Giorgio Rizzo
* * * * * *

Supervisors
Prof. E. Macii, Supervisor
Prof. A. Calimera, Co-supervisor

Doctoral Examination Committee:
Prof. Alberto Nannarelli, Referee, DTU - Danmarks Tekniske Universite
Prof. Michele Magno, Referee, Eidgenössische Technische Hochschule Zürich
Prof. Andrea Acquaviva, Università di Bologna
Prof. Silvia Chiusano, Politecnico di Torino
Prof. Paolo Garza, Politecnico di Torino

Politecnico di Torino
July 16, 2019

I hereby declare that, the contents and organisation of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

. . . . . .

Roberto Giorgio Rizzo
Turin, July 16, 2019

Summary

Energy efficiency has become the main constraint for most of today's information and communication technologies, from those involving high-performance computing (e.g., cloud services) to those deployed on low-power applications (e.g., portable systems for the Internet-of-Things). In the past decades, the pursuit of energy efficiency was mainly supported through the advance of the underlying CMOS technology. Moving towards a new node was the guarantee to achieve more than 90% of energy savings. However, as soon as the CMOS entered the nanometric regime, improvements brought by a technology shift have shrunk substantially, reaching 20% and then further less generation by generation. To make matters worse, production costs raised dramatically due to the technological impediments imposed by physical geometries below the 28 nm mark. This made technology scaling impractical for many cost-sensitive applications.

New sophisticated energy-aware design practices were then introduced to alleviate the suffering of a slow technology scaling. Very soon, low-power and energy-management techniques become the actual kernel of any design and optimization flow. Unfortunately, also design techniques are not fully renewable, namely, their effectiveness degrades with the advance of the technology nodes. This is the case of voltage scaling, for instance, which encountered the 1.0 V plateau that still holds today, but also other architectural-level techniques, such as multi-core/many-core solutions, which have been seriously limited by stringent dark-silicon constraints.

The end of Moore's law is not just a technology issue; it is also the prelude of a design crisis that will soon require to rethink the optimization and integration strategy of digital circuits and systems. A radical solution to all these concerns has to come yet. However, the recent growth of data-centric applications is opening to new design paradigms that alleviate the pressure. Much room is at the application-level indeed, where alternative energy-management knobs are available. The basic idea is that of integrating the quality-of-results as a new dimension in the design space. Leveraging the intrinsic error-resilience of data-centric applications, it is thereby possible to implement an *Energy-Accuracy Scaling* (EAS) which is orthogonal to the technology adopted and the low-power design strategy deployed. At the basis of this concept, there is the simple intuition that an application whose output can be degraded without affecting the quality perceived by the user may require lower energy consumption for the same amount of work.

The broad objective of this dissertation is to introduce advanced design solutions that improve the approach the EAS paradigm is implemented. Two new strategies are presented which reduce the design overhead of classical approximate solutions; according to the revisited taxonomy introduced in this thesis, one of the proposed strategies belongs to the class of *Adaptive* EAS, while the second falls under the label of *Static* EAS. With *Adaptive* EAS, the optimal energy-accuracy tradeoff is achieved by measuring some quality metrics directly on-chip, at run-time, establishing a feedback loop that drives the energy minimization. These metrics can be obtained by explicitly measuring the output accuracy, or by indirect measurements, e.g., through the output error rate. With *Static* EAS, the energy-accuracy tradeoff is fixed at design-time by functional speculation, i.e., a modification of the logic functionality through algorithmic or circuit simplifications which induce energy savings for a worst-case accuracy loss.

The Adaptive solution encompasses the extension of the conventional Error Detection-Correction techniques for data-driven voltage scaling in order to trade system accuracy for energy reduction. The new mechanism, called *Approximate Error Detection-Correction* (AED-C), is built upon in-situ *elastic timing monitors* which allow to implement a lightweight error management scheme.The AED-C implements EAS using the error detection coverage as a knob: a low error coverage accelerates supply voltage over-scaling thus to achieve more significant energy savings at the cost of quality-of-result; a high error coverage lessens the voltage scaling leading to higher accuracy at the cost of lower energy savings. As EAS does not have to ensure full error coverage, the traditionally large area/energy overhead of conventional techniques is drastically reduced. Simulations over a representative set of applications/circuits, e.g., Multiply-Accumulate (MAC) unit, Discrete Cosine Transform (DCT), FIR and IIR filters, provide a comparative analysis with the state-of-the-art techniques. The collected results show that AED-C substantially reduces the average energy-per-operation and the area overhead, still guaranteeing reasonable accuracy.

The static EAS strategy, instead, is developed exploiting Machine Learning theories which suggest alternative forms to represent relationships among data. Such theories find their application in the Boolean domain, where logic functions can be described as inference rules. The novel paradigm, named as *Inferential Logic*, leverages the concept of statistical inference for the design of combinational logic circuits that are able to mimic Boolean functions to a certain degree of accuracy. These inferential logic circuits run *quasi-exact* computations trading energy efficiency for accuracy in error-resilient applications. The figures-of-merit of an *Inferential Multiplier* are quantified using representative image processing applications as a case study. A comparative analysis against a state-of-the-art Booth Multiplier proves the inferential logic representation simplifies the circuit complexity reducing the overall area/delay. As a result, the inferential multiplier can exploit latency reduction for power optimization guaranteeing a fixed average accuracy.

Acknowledgements

I am very much indebted to my supervisor Prof. Enrico Macii, for giving me the opportunity to work in the EDA group of Politecnico Di Torino.

I particularly would like to express my deep sense of gratitude to my mentor Prof. Andrea Calimera, for its valuable guidance and experience sharing. It was its encouragement that motivated me to explore new dimensions in my research.

I would like to acknowledge A*STAR Singapore - Institute of Microelectronics to allow me to join the "Integrated Circuits & Systems - Digital IC" research group for eighteen months of my PhD career. In particular, I would be grateful to Prof. Zhou Jun for its support and knowledge sharing.

I gratefully acknowledge the project "SENSEI - Sensemaking for Scalable IoT Platforms with In-Situ Data-Analytics: A Software-to-Silicon Solution for Energy-Efficient MachineLearning on Chip," funded by Compagnia di San Paolo, with Università di Bologna as academic partner and ST Microelectronics as non-academic partner.

To Elena, for her patience and love.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Context & Motivations

Energy efficiency is a critical constraint for most of today's applications spanning a broad range of performance and form factor, from high-performance computing (e.g., cloud servers and GPUs) to low-power applications (e.g., edge IoT). The available energy budget directly affects the design and optimization of digital Integrated Circuit (ICs) determining the achievable performance.

In the last few decades, energy reduction has been driven by technology scaling, guaranteeing more than 90% of the energy savings at each new CMOS generation [75]. Figure 1.1 shows that energy consumption for general-purpose processors (blue dashed plot) has dropped at the rate of 100×/decade, as theorized by the Koomey's law [75]. A similar trend, also known as Gene's Law, has been observed for Digital Signal Processors (DSPs - green dashed plot) [66]. However, in the last two decades, while GPUs energy improved by 12-20×/decade (red dashed plot), the energy reduction for processors trend has significantly slowed down to 10×/decade (black dashed plot).

Also, with the end of Moore's law, pure technology advances are expected to have less impact on energy minimization, only 4× in the decade ahead (dotted black line), as the new CMOS generations will be able to guarantee savings lower than 20% at each new technology node, with a limited set of remaining generations [60]. Besides, the transistor cost below 28 nm is no longer being optimized [140], making further technology scaling an unfeasible option for cost-sensitive applications, e.g., sensor nodes in IoT. In other words, the need for energy-efficient design practices at different levels of abstraction have become more and more critical as the technology scaling alone cannot guarantee the energy savings brought with previous nodes.

Likewise, standard design-level energy optimization strategies find lower margins of applicability. Among them, the Voltage Scaling technique, extensively leveraged at circuit-level in the last two decades, which encountered the 1.0 V plateau that still holds today. Moreover, as commercial low-power processors and standard cell libraries voltage approaches transistor threshold, with 0.6 V corner already state-of-the-art, limited

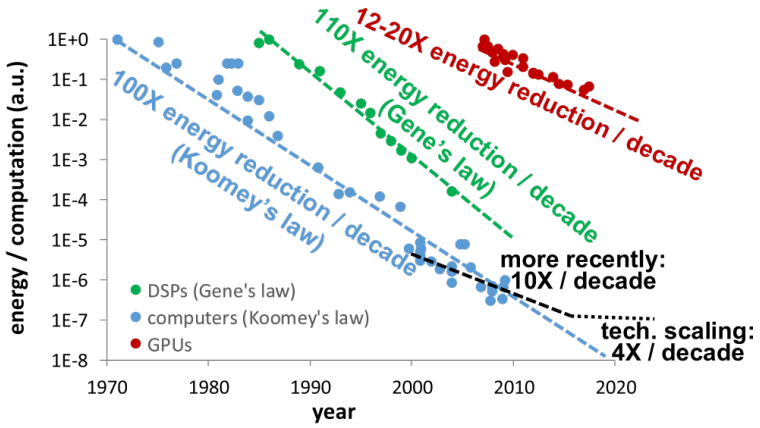


Figure 1.1: Energy per computation trend vs. year for general-purpose computer, DSPs and GPUs [3].

room for further voltage scaling has been left [35]. At architectural-level, the energy savings achieved through to parallelism have lessened, especially in those applications whose workload is not naturally parallelizable. For example, in portable electronics platforms (e.g., smartphones) the number of simultaneously active cores has settled; more cores are added only to cover a broader range of energy-performance tradeoffs.

In summary, the historical energy reduction trend in ICs cannot be solely supported by unceasing technology scaling, more aggressive voltage scaling or deeply parallelized architectures [22]. This is a prelude of a design crisis that will soon require to rethink the optimization and integration strategy of digital circuits and systems. A radical solution to all these concerns has to come yet. However, the literature offers a solution. The recent growth of data-centric applications is opening to new design paradigms that alleviate the pressure. Much room is at the application-level indeed, where alternative energy-management strategies are available. They propose to integrate the system accuracy as a new dimension into the design/optimization space to have a further knob for adjusting circuits energy-efficiency [1] [3]. Leveraging the intrinsic error-resilience of data-centric applications, it is thereby possible to implement an *Energy-Accuracy Scaling* (EAS) which is orthogonal to the technology adopted and the low-power design strategy deployed. At the basis of this concept, there is the simple intuition that an application whose output can be degraded without affecting the quality perceived by the user may require lower energy consumptions for the same amount of work.

The benefits of EAS-based design strategy can be proved by an explanatory example of a smartphone delivering a video stream [1]. The accuracy of the output video can be tuned to lower values for saving energy, whenever *(i)* the quality of lighting is poor (as sensed by a light sensor); *(ii)* the battery is drained out, and extending its lifetime becomes a priority; *(iii)* the user is on the go (as sensed by motion sensor) and is not focusing on the content. Hence, integrated sensors might be used to understand the context

scenario at run-time and drive the device energy efficiency through the power management system. From a power policy viewpoint, a possible approach might reduce the energy/frame through an aggressive voltage over-scaling targeted to a minimum value of video quality that does not compromise the user experience (e.g., tiny pixels imperfections due to occasional timing faults). The accuracy target, which can be managed by directly measuring the output quality or by an indirect estimation of the accuracy value, is used as a knob for generating the appropriate energy-accuracy tradeoff for the systems involved in processing/visualizing the contents. Similarly, several multimedia applications, e.g., audio/image processing, can be considered energy-accuracy scalable systems as their output quality can be traded for energy savings.

The above illustrative case reports an EAS strategy which integrates the sensed context information and the accuracy measure in a feedback loop used as a knob to adjust the energy savings adaptively. However, as discussed in this dissertation, the energy-accuracy tradeoff can also be statically fixed at design-time or can be set dynamically among different energy-accuracy operating points pre-defined at worst-case conditions, thus, ensuring acceptable quality degradation levels for specific energy savings.

Another motivating example refers to an emerging yet very promising market, i.e., the Internet-of-Thing (IoT). The key to success for the IoT galaxy is the availability of always-on smart objects with embedded ICs that can process/transmit sensor data ceaseless. Due to the limited budget of energy made available by small batteries, such ICs must show ultra-low power consumption thus to guarantee reasonable throughput. Thus, the need for energy efficiency imposes stringent design constraints on circuits and systems that cannot be achieved with classical low-power techniques, such as Dynamic Voltage and Frequency Scaling, Clock-Gating, Power-Gating. The literature is plenty of viable options to match these stringent energy constraints [2]. However, as described above[1] [3], the recent trends in EAS highlight the rise of aggressive power-management strategies that leverage the error tolerance of specific IoT applications, e.g., video/audio sensor node or wearable trackers, to trade energy efficiency for quality-of-results through mechanisms that give real-time feedback, e.g., environmental conditions or user status. Such information is used by the power management system to identify and deliver the most appropriate energy-accuracy tradeoff.

1.2 Objectives & Contributions

The broad objective of this dissertation is to provide a new taxonomy and classification of *Energy-Accuracy Scaling* (EAS) techniques, supported by a detailed pros&cons analysis. Besides, two novel strategies which aim at implementing low overhead EAS are proposed. According to the presented taxonomy, these strategies belong to the *Adaptive* and *Static* classes of EAS. In the *Adaptive* EAS methods the optimal energy-accuracy tradeoff is achieved by measuring quality metrics directly on-chip, at run-time, establishing a circuit compliance feedback loop that drives the energy minimization.

These metrics can be obtained by explicitly measuring the output accuracy, or by indirect measurement of the accuracy value, e.g., through the output error rate. On the contrary, in the *Static* approaches, the energy-accuracy tradeoff is fixed at design-time by functional speculation, i.e., a modification of the logic functionality through algorithmic simplifications, or circuit- and gate-level approximations, which induce energy savings for a worst-case accuracy loss.

A detailed discussion on the main contributions of this manuscript follows below.

Approximate Error Detection-Correction (AED-C) for Adaptive EAS

The contributions of this dissertation on the *Adaptive* solutions concern the extension of the conventional Error Detection-Correction techniques [39] for data-driven voltage scaling in order to trade system accuracy for energy reduction. This mechanism, called *Approximate Error Detection-Correction* (AED-C), is built upon in-situ *elastic timing monitors* which enable an error management scheme suited to adaptive power management (e.g., Adaptive Voltage Over-Scaling) on error-resilient applications. Inspired by the working principle of Approximate Computing, AED-C implements EAS using the error detection coverage as a knob: a low error coverage accelerates supply voltage over-scaling thus to achieve larger energy savings at the cost of quality-of-result; a high error coverage lessens the voltage scaling leading to higher accuracy at the cost of lower energy savings.

As EAS does not have to ensure full error coverage, the traditionally significant area/energy overhead of the conventional error detection-correction techniques may be reduced by using a simpler error management circuitry. Simulations over a representative set of applications/circuits, e.g., Multiply-Accumulate (MAC) unit, Discrete Cosine Transform (DCT), FIR and IIR filters, provide a comparative analysis with the state-of-the-art techniques. The collected results show that AED-C substantially reduces the average energy-per-operation (up to 44.7% savings w.r.t. Razor-driven Adaptive Voltage Over-Scaling) and the area overhead (3.3% vs. 62.0%), still guaranteeing reasonable accuracy. As an example, when applied to a real-life image processing application, i.e., Discrete Cosine Transform Unit (DCT) integrated into a JPEG compressor, AED-C shows 51.9% energy savings (w.r.t. a baseline DCT implementation) and a PSNR of 48.45 dB (w.r.t. baseline JPEG images)

The strength of adaptive strategies lies under the capability of designing techniques that can support context-driven reconfiguration/adaptation enabling low-power knobs to be tuned at a finer scale granularity. However, such efficiency poses many challenges that need to be coherently addressed at different levels of abstraction, from the application-/software-level down to the circuit-level and power delivery system. The research key aspects that characterized this contribution concern:

1. the efficient implementation of in-situ timing sensors;

2. the design of architectural solutions responsive to the timing sensors real-time

feedback;

3. the integration of voltage-accuracy scaling policies that drive the power delivery system;

4. the development of EDA tools which implement a new across-level optimization/simulation approach, as preliminary design exploration, chip design and verification, to meet performance and energy-accuracy constraints.

Low-power Infential Logic for Static EAS

Within the scope of *static* EAS, a new strategy is developed exploiting Machine Learning (ML) theories which suggest alternative forms to represent relationships among data. Such theories find their application in the Boolean domain, where logic functions can be described as inference rules. This novel paradigm, know as *Inferential Logic* [146], leverages the concept of statistical inference for the design of combinational logic circuits that are able to mimic Boolean functions to a certain degree of accuracy. Although several options for building abstract models are available, *Classification Trees* have proven to achieve high accuracy with a low level of complexity in many application cases [15].

Inferential logic circuits infer output values by evaluating the key features of the function learned during the training stage, skipping the logic operation. Their intrinsically speculative nature lead to low circuit complexity at the cost of a quality-of-results drop. In other words, inferential logic circuits run *quasi-exact* computations trading energy efficiency for accuracy loss in error-resilient applications. This principle has a positive impact on arithmetic applications [145] [147] where inferential circuits design can lead to architectures more prone to support aggressive adaptive power management. For this reason, the figures-of-merit of an *Inferential Multiplier* are quantified using representative image processing applications as a case study. A comparative analysis against a state-of-the-art Booth Multiplier proves the inferential logic representation simplifies the circuit complexity reducing the area by 22%. Also, the inferential multiplier can exploit 2× latency reduction for power optimization guaranteeing 76% average accuracy.

The research key aspects of this contribution focused on:

1. the formulation of theories and methods that could support the involvement of ML algorithms within ICs design.

2. the development of ML-based EDA tools for inferential logic circuits, i.e., the integration into the commercial platform of design flow steps that map logic functions to statistical inference circuits based on Classification Trees.

3. the implementation of tools for inferential circuits design exploration, simulation, and verification.

1.2.1 Main Research Achievements

The novel techniques presented in this dissertation have been conceived to improve existing EAS schemes, in terms of efficiency and flexibility, by matching the following requirements/constraints: *(i)* do not apply any "irreversible" modification of the circuit (i.e., re-synthesis stages); *(ii)* introduce bounded design overhead (area/power); *(iii)* avoid dedicated optimization algorithms that may be hard to integrate into industrial design kits. The research achievements of the proposed strategies can be summarized as follows:

- **Approximate Error Detection-Correction (AED-C) for Adaptive Energy-Accuracy Scaling (EAS)**:
  porting of the Approximate Computing concept to Adaptive EAS techniques using the timing faults coverage as a knob to trade energy for quality-of-results. AED-C offers across-level contributions in EDA methodology/tools, architectural solutions, and circuit implementation.

- **Low-power Inferential Logic for Static Energy-Accuracy Scaling**:
  porting of Machine Learning (ML) theories into the Boolean domain to implement logic functions described as statistical inference rules. The inferential logic circuits obtained through the proposed ML-based design flow present architectures prone to support aggressive Voltage (Frequency) scaling in exchange for a certain degree of accuracy.

1.3 Dissertation Outline

After this brief introduction of the context scenario and the main challenges/contributions of this dissertation, *Chapter 2* provides the reader with an overview of the notions, approaches, and methods for Energy Scaling. Firstly, the chapter deals with the classical power optimization strategies proposed in the literature in the last two decades and mostly already integrated into commercial tools. Follows a brief review of the techniques that ensured a more efficient energy reduction by delivering dynamic management of the power/frequency (i.e., DVS and DVFS). The last part chapter focuses on the Better-Than-Worst-Case design strategy and how this principle enables an adaptive, i.e., context-/data-driven, energy scaling yet guaranteeing always-correct computation, i.e., preserving the quality-of-results. An overview of techniques leveraging Better-Than-Worst-Case design is proposed with particular attention to those based on the Timing Speculation principle and implemented through timing errors monitors.

Chapter 3 discusses the strategies conceived to operate beyond the Energy scaling through the integration of the quality-of-results as a new dimension in the design/optimization space. The main techniques exploiting this concept, known as Energy-Accuracy Scaling (EAS), are classified in a Taxonomy. The three identified classes, *Static*, *Dynamic*

and *Adaptive* EAS, are exhaustively disclosed through a pros&cons analysis. This represents a preliminary contribution to this work.

The core of this dissertation is the development of two novel approaches which aim at improving the techniques that belong, respectively, to the Static and Adaptive class. In *Chapter 4*, the proposed Adaptive EAS solution is disclosed along with the stages that bring standard Error Detection-Correction techniques to be more prone for aggressive power management like Adaptive Voltage Over-Scaling (AVOS). These stages include the development of in-situ elastic timing sensors that enable the Approximate Error Detection-Correction (AED-C) strategy which plays with the timing faults coverage knob to achieve fine-tuned energy-accuracy efficiency. *Chapter 5* focuses on the development of a Static EAS solution based on Inferential Logic. A Classification Tree training step is integrated into the standard design flow to implement an Inferential Multiplier. A comparative analysis with a state-of-art Booth multiplier discloses the benefits of the proposed EAS solution. *Chapter 6* provides final considerations and future works.

Chapter 2

Energy Scaling through Low-Power Optimization

2.1 Power Consumption Model in CMOS Technology

The continuous progress in microelectronic circuits has been maintained mostly by CMOS technology scaling, which results in exponential growth both in device density and performance as predicted by Moore's law [105]. The development of EDA tools and the effort of digital system designer to develop novel computing architectures had accelerated this process and had led to more complex and high-performance circuits. However, as the technology scaling enters nanometer regime, energy efficiency became one of the main design concerns for today's digital Integrated Circuits (ICs). This is true not just for portable applications (e.g., smartphones, wearables, IoT galaxy devices), where the energy budget is limited by the use of thin batteries [11, 2], but also for high-performance applications (e.g., cloud servers and GPUs) where a proper resource management improves sustainability at a large scale [54, 163].

In the last two decades, in the semiconductor industry, energy consumption has been mainly tackled by setting power dissipation as a primary design constraints. Reducing power can extend battery lifetime of portable systems, decrease cooling costs, as well as increase system reliability. In a digital CMOS circuit, the two major contributions to power dissipation have a twofold nature, *Dynamic* and *Static*:

$$P = P_{dynamic} + P_{static} \tag{2.1}$$

Dynamic power originates by the run-time activity of the circuit, i.e., by the input workload that is fed to combinational logic and sequential cells like Flip-flops and Latches. *Static* power, on the contrary, refers to the power dissipated in stationary condition, i.e., when there is no circuit activity. The following sections explain in details the sources of both power components.

Dynamic Power Sources

Two major sources of dynamic power consumption can be identified: *Switching* and *Short-circuit* power:

$$P_{dynamic} = P_{sw} + P_{sc} \tag{2.2}$$

The **Switching power** P_{sw}, is due to to charging and discharging the capacitance driven by the circuit, i.e., load capacitance, wire capacitance and the gate self-capacitance. Figure 2.2 depicts all the capacitances seen by a CMOS inverter. current and charging/discharging of capacitances For a CMOS gate working in a synchronous environment, P_{sw} can be modeled by the following equation:

$$P_{sw} = \frac{1}{2} \cdot C_L \cdot Vdd^2 \cdot F_{clk} \cdot E_{sw} \tag{2.3}$$

where, C_L is the output load capacitance of a gate; it depends on the physical parameters of gates and wires. Vdd and F_{clk} are the supply voltage and and the clock frequency; they are design parameters as they affect the speed of the system. E_{sw} takes into account the switching activity of the gate, defined as the probability of the gate's output to make a logic transition during one clock cycle; this terms models the case that, except for the clock tree network cells, gates might not make a transition at each clock cycle. According to Equation 2.3, reductions of P_{sw} can be achieved by: *(i)* Supply Voltage (Vdd) Scaling. This allows P_{sw} to scale quadratically; this comes with circuit performance decrease (lower circuit speed). A simple model of the propagation delay of a cell is reported in Equation 2.1.

$$D_p \propto C_L \cdot \frac{Vdd}{(Vdd - V_{th})^{\alpha}} \tag{2.4}$$

As to compensate the circuit performance loss introduced by reducing voltage, speed optimization is applied first, followed by supply voltage scaling; this strategy brings the design back to its original timing, but with a lower power requirement [122]. With *(ii)* Frequency (F_{clk}) Scaling , P_{sw} can be traded for circuit speed; power, in fact, drops linearly. for lower circuit speed. Selective frequency scaling (as well as voltage scaling) can be applied to specific units with no penalty in the overall system speed [122]. Finally, *(iii)* P_{sw} can be reduced by the minimization of Switched Capacitance, defined as $C_{sw} = C_L \cdot E_{sw}$; this approach has a lower impact on performance, yet allowing significant power savings [122]. Static solutions (i.e., at design time) handle switched capacitance minimization through area optimization (that corresponds to a decrease in the capacitive load) and switching activity reduction via exploitation of different kinds of signal correlations (temporal, spatial, spatio-temporal); Dynamic techniques, on the other hand, aim at eliminating power wastes that may be originated by the application of certain system workloads (i.e., the data being processed).

The **Short-circuit Power** P_{sc}, is strictly related to the complementary nature of CMOS logic. In actual designs, the assumption of the zero rise and fall times of the

input wave forms is not correct. The finite slope of the input signal causes a direct current path between Vdd and Gnd sources for a short period of time during switching, while the NMOS and the PMOS transistors are simultaneously active. Thus, a current flows through CMOS Pull-up and Pull-down stages while cell switches. Usually, the contribution of P_{sc} to the total power is lower than P_{sw}, yet optimizations can be applied to it. In particular, transistors sizing is fundamental, as well as the relation between the transition time of input signal and output signal. As reported in [122], the matching between transition times of input and output signals is a rule of thumb which allows the overall short-circuit current to be minimized.

Static Power Sources

The Static (or Leakage) Power dissipation of a circuit can be expressed as follows:

$$P_{static} = I_{static} \cdot Vdd \tag{2.5}$$

where I_{static} is the current that flows between the power supply rails in the absence of switching activity. Ideally, the static power consumption of a CMOS circuit should be zero, as no static current should flows between PMOS and NMOS devices. However, in non-ideal devices a leakage current flows through the reverse-biased diode junctions of the transistor; I_{static} can be caused by different physical phenomena [129] that are modeled as three main current contributions. *(i) Gate-oxide leakage* due to the reduction of the gate oxide thickness as a consequence of technology scaling. The electrical field across the oxide increases, thus a leakage current due to tunneling effect is generated. *(ii) pn-Junction Reverse-Bias Current* due to the reverse biasing condition of drain and source to well junctions. *(iii) Subthreshold leakage* is current flowing between drain and source when the gate voltage is under threshold ($V_{GS} < V_{th}$). In fact, a non-ideal MOS device it is not an ideal switch controlled by the V_{GS}.

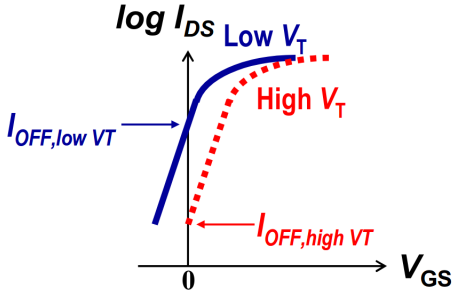


Figure 2.1: $I_{DS}(V_{GS})$ for different values of V_{th}.

As reported in Figure 2.1, when V_{GS} is fixed to zero, the lower the threshold voltage V_{th}), the higher the drain-source current I_{DS} value. This translates into higher Static Power consumption. The Equation 2.1 suggests that the speed of the gates can be increased by reducing the V_{th}); however, as mentioned earlier, the price to pay is larger

Static Power.

To be noticed that Leakage current strictly depends on temperature. Hence, the operating temperature of the chip must be kept below the thermal safe temperature, T_{safe}, to ensure that the chip operates within the safe range. This can be achieved by a well-designed packaging, a physical-design stage aware of the temperature issues, and by reducing the power consumption.

2.2 Classical Low-Power Optimization Techniques

Starting from the CMOS power model, different techniques at various level of abstraction have been proposed in order to minimize power consumption. Most of them focus on reducing the dynamic power of digital circuits from the architectural-level down to individual logic-gates and transistor-level. Figure 2.2 report a taxonomy of the most representative low-power optimization methodologies, whose details are discussed in the following sections.

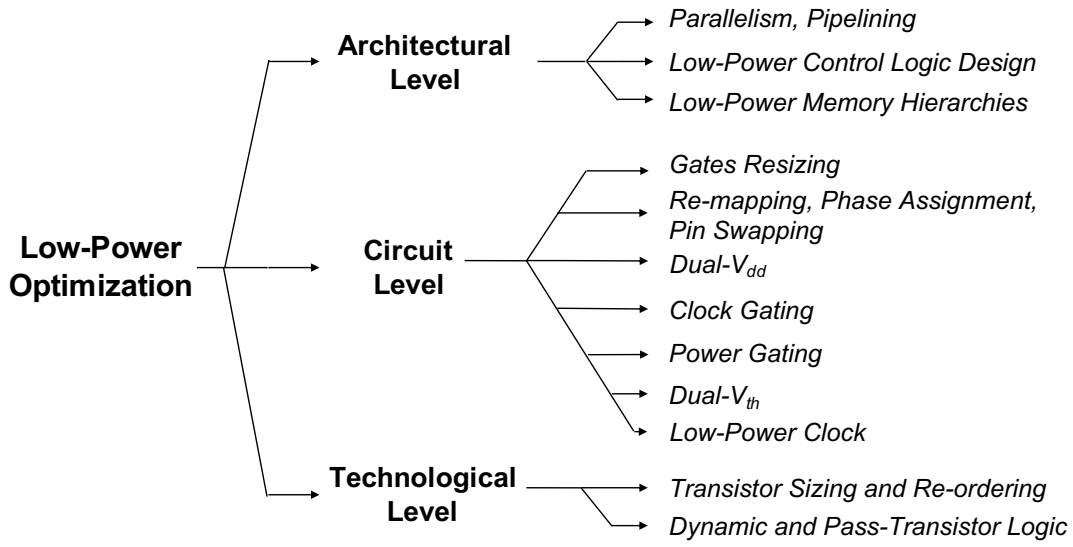


Figure 2.2: Taxonomy of Classical Low-Power Optimization Techniques.

2.2.1 Architectural Level

Power efficiency can be obtained at the architectural level once the data-path, memory, control and interconnect structures are fully defined. This section describes some of the historical architectural low-power techniques. Optimization at this level of abstraction means finding the best architectural composition for a given algorithm.

Parallelism and Pipelining

Parallelism and *Pipelining* [121] approaches optimize the dynamic power ensuring the same throughput at the cost of area overhead. Specifically, *(i)* the system is sped-up by architecture re-organization, then *(ii)* the available timing slack can be consumed through supply voltage scaling.

Parallel Architecture: in this approach a functional unit working at frequency F_{clk1} is replaced by two equivalent functional units operating at $F_{clk2} = \frac{1}{2} \cdot F_{clk1}$, respectively, on the rising and falling edge of the clock. The timing slack obtained by increasing the clock period can be used to reduce the supply voltage. However, this method presents two main drawbacks: *(i)* the area is doubled with a consequent increase of leakage power consumption; *(ii)* it presents a scalability issue as, when the supply voltage approaches the threshold voltage (V_{th}), the delay of the circuit increases so quickly that which in turn may increase the overall energy consumption.

Pipelined Architecture: the propagation delay (D_p) of a synchronous logic circuit can be reduced by splitting it combinational paths into several portions. As each portion presents a worst-case delay lower than D_p, the supply voltage can be scaled to consume the available slack. Similarly to the parallel architecture technique, this technique comes with some drawbacks: *(i)* overall latency increase, *(ii)* die area overhead and *(iii)* scalability issues.

The combination of pipelining and parallelism can result in further power reduction as the supply voltage can be reduced more aggressively [121].

Low-Power Control Logic Design

Within a Processor Architecture, the control logic that manages the operations is a finite state machine (FSM): for each state, a specific piece of circuitry is activated. Thus, low-power control logic can be designed by optimizing FSMs for power reduction. As an example, FSM optimizations can be achieved by *(ii)* encoding the FSM states to minimize the switching activity or by *(ii)* decomposition of FSM into sub-FSMs, where only the circuitry needed for the currently executing sub-FSM is activated. According to [46], applying these techniques jointly reduces processor power from 30-90%, while area raises from 20-120%.

Low-Power Memory Hierarchies

As for the previous approach, this technique applies to Processor Architecture, specifically to the Memory component. Power consumption in memory can be bounded in two ways: either by reducing the power dissipated in memory access, or by reducing the number of accesses to memory [156].

Concerning the former strategy, the literature is plenty of optimization techniques. An effective way to reduce power in memory access is obtained by partitioning the memory into smaller sub-banks. This can be done by splitting memory into smaller,

independently accessible components at different granularity such that for each memory access only the required circuitry is activated. Implementing a combination of sub-banking, multiple line buffers and bit-line segmentation might lead to 75% on-chip cache power reduction [49]; this approach is technology-independent and does not compromise the processor cycle time.

Another well-known strategy to save memory access power is implemented by augmenting the memory hierarchy with specialized cache structures; such a memory organization allows to decreases memory accesses. A simple implementation of this principle is achieved by using a trace cache. This system keeps track of the instructions in their executed order rather than their compiled one. Hence, if an instruction sequence is already stored in the trace cache, it does not require to be fetched from the instruction cache and can be decoded directly from the trace cache [59]. Nevertheless, conventional trace caches (CTC) may lead to a power increase in the fetch unit. Simultaneous access to both the trace cache and the instruction cache may occur, in fact. Dynamic direction prediction-based trace cache (DPTC) avoids simultaneous accesses to the trace cache and the instruction cache achieving 38.5% power reduction over CTC, while only trading a 1.8% performance overhead w.r.t. CTC [59].

2.2.2 Circuit Level

In the last two decades, Logic-level power optimization strategies have been extensively researched [142]. The most powerful of these techniques have been made available into EDA commercial logic synthesis tools enabling logic-level power optimization not only for structured logic in large-volume components, as microprocessors (e.g., functional units in the data-path), but, also, for unstructured logic and low-volume VLSI circuits.

A more detailed description of logic-level optimization techniques follows. To be noticed that, as for architectural-level techniques, power is never the only cost metric to take into account during optimization. Performance/area are tightly constrained as well [101].

Gates Resizing

A common strategy of power optimization lies in a simple rule: *a logic network can be transformed to minimize power only if the critical path length is not increased.* Under this hypothesis, an effective technique is based on *path equalization.* A wide distribution of path delays characterizes logic circuits, thus path equalization can be obtained by *gate resizing*. Resizing focuses on fast combinational paths. Gates on fast paths are down-sized, thereby decreasing their input capacitances, while at the same time slowing down signal propagation. By slowing down fast paths, propagation delays are equalized, and power is reduced by joint spurious switching and capacitance reduction. Resizing does not always imply down-sizing. Power can also be reduced by enlarging (or buffering)

heavily loaded gates, to increase their output slew rates. Fast transitions minimize short-circuit power of the gates in the fan-out of the gate which has been sized up, but its input capacitance is increased. In most cases, resizing is a complex optimization problem involving a tradeoff between output switching power and internal short-circuit power on several gates at the same time.

For arithmetic circuits, such as adders of multipliers, path equalization ensures that signal propagation from inputs to outputs of a logic network follows paths of similar length. When paths are equalized, most gates have aligned transitions at their inputs, thereby minimizing spurious switching activity (which is created by misaligned input transitions).

Usually, during this logic optimization, technology parameters such as supply voltage are fixed, so the degrees of freedom are exploited in the gate resizing of a given logic netlist.

Re-mapping, Phase Assignment, Pin Swapping

Logic-level power minimization techniques as Re-mapping, Phase Assignment and Pin Swapping, can be classified as *local transformations*. They are applied to gate netlists, and focus on nets with large switched capacitance. Most of these techniques replace a gate, or a small group of gates, around the target net, in an effort to reduce capacitance and switching activity. Similarly to gate resizing, local transformations must carefully balance short circuit and output power consumption.

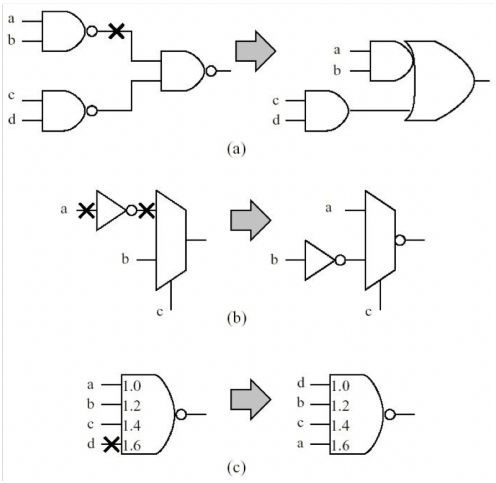


Figure 2.3: Local transformations: (a) re-mapping, (b) phase assignment, (c) pin swapping [Courtesy of STMicroelectronics].

Figure 2.3 shows three examples of local transformations. In (a) Re-mapping transformation is reported: a high-activity node (marked with x) is removed thanks to a new mapping onto an AND-OR gate. Phase Assignment (b) is exploited to remove one

of the two high-activity nets marked with x. Pin swapping (c) is applied to connect a high-activity net to the input pin of the 4-input NAND with the minimum input capacitance..

Dual-Vdd

Path delay equalization can be obtained by playing with supply voltage Vdd. In a logic circuit, a vast majority of gates do not belong to critical paths. Thus, gates on fast paths can be fed with lower Vdd, thereby decreasing their switching power (quadratically) while at the same time slowing down signal propagation. Iteratively, slack available on fast paths can be consumed by feeding gates belonging to non-critical paths with lower Vdd. Due to the complexity of on-chip voltage regulators and power delivery network, the number of supply voltages to be assigned should remain low. In literature, two values of Vdd are considered to be the best trade-off; from here Dual-Vdd.

Dual-Vdd Algorithms as [152] [34] [87] [83] have been proposed for automatic Vdd assignment. They are based on heuristic methods that perform iterative graph visit that iteratively trying to find the optimal netlist partitioning which minimizes power consumption satisfying the timing constraints. The result of these algorithms is a logic circuit mapped to *voltage islands*.

The major drawback of Dual-Vdd assignment is due to the employment of level shifters necessary to interconnect tiles of die fed at different voltages. This generates overhead and complexity in the physical design.

Clock Gating

Clock gating relies upon an intuition: *the output of a logic block is not always useful*. Thus, the power dissipated by the switching activity can be reduced by disabling the clock signal to unused blocks. Clock gating provides a way to stop the clock selectively, and thus force the original circuit to make no transition, whenever the computation to be carried out by a hardware block at the next clock cycle is useless. In other words, the clock signal is disabled following the idle conditions of the block.

The main effect of clock gating is the reduction of the switching power of the combinational logic fed by the gated registers. Also, clock-tree and registers power dissipation decrease.

As reported in [67], many implementations of clock gating have been proposed. The most common is Latch-Based clock gating, where the clock enable flag is stored in a latch and *ANDed* to the clock signal. The latch is used to avoid hazard propagation and to reduce the degradation of clock signal rising and falling transitions.

Power Gating

Power Gating strategy has been conceived for Static Power reduction. The main idea behind this technique is to limit leakage current of gates in idle conditions [62]. As

many works in literature report [7] [18] [19], Power gating is mostly implemented by inserting high-V_{th} transistors, namely *sleep transistors*, in series with the CMOS pull-up and/or pull-down networks. When sleep transistors are *off*, the sub-threshold current of PMOS/NMOS networks is reduced; when they are *on*, the logic circuit should keep operating in nominal condition, thus, sleep transistors have to be properly sized to reduce the overall circuit delay penalty.

The insertion of sleep transistors is a complex design task: it requires optimization algorithms for *(i)* adjusting cell clusters, *(ii)* sleep transistors sizing and *(iii)* the distribution network of *on/off* signals.

Dual-V_{th}

As for Power Gating strategy, Dual-V_{th} aims at Static Power reduction. As reported in literature [58] [108], Dual-V_{th} can be implemented using multi-V_{th} cells. This is made possible by the fact that silicon vendors provide technological libraries with *(i)* high-V_{th} cells, fast but having high leakage power, and *(ii)* low-V_{th} cells, slow but less leaky, which can be integrated into the same technological process The basic idea of Dual-V_{th} is based on the principle behind Dual-Vdd, i.e., *path equalization*. The design is firstly synthesized and mapped onto low-V_{th} cells; then the gates belonging to non-critical paths are replaced with high-V_{th} cells as long as the timing constraints are satisfied. Usually, Dual-V_{th} is applied simultaneously to gate resizing in order to reduce the total power consumption.

Low-power Clock

Classical examples of low-power clocks are *half-frequency* and *half-swing* clocks. Such clock signals reduce frequency and voltage respectively. Traditionally, hardware events such as register file writing occur on a rising clock edge. Half-frequency clock run at half the speed of regular clock and synchronize events using both edges halving the clock switching power. Reduced-swing clocks use lower voltage signal reducing power quadratically [73]. Data metastability issues should be addressed appropriately applying this solution.

2.2.3 Technological Level

In this section, the CMOS power optimization strategies go down to technological solutions. Three main approaches are considered: *Transistor Sizing and Re-ordering*, *Low-power Clock* and *Pass-Transistor Logic*. Similarly, to architectural-/logic-level optimization, these techniques aim at finding the best tradeoff between performance and power.

Transistor Sizing and Re-ordering

Combinational cells transistor sizing can have a significant impact on circuit delay and power dissipation. If the size of the transistors in a cell increases, the delay of the cell decreases. This because transistors with larger gate widths drain more current than smaller transistors. Unfortunately, larger transistors also contribute more device capacitance to the circuit and, consequently, result in higher power dissipation. Also, larger transistors suffer more severe short-circuit currents, which should be avoided whenever possible. The transistor sizing also affects the delay of a circuit: as gate size increases, the load capacitance of the cell increases leading the delay of the fan-in gates to raise. Hence, given a delay constraint, finding an appropriate sizing of transistors that minimizes power dissipation is a computationally difficult problem.

An efficient low-power strategy minimizes transistor size whenever possible to reduce the dynamic power while meeting performance constraints. Thus, the transistors that lie away from the critical paths of a circuit are usually the best candidates for down-sizing.

A typical class of heuristic algorithms associates with each transistor a tolerable delay, which varies depending on how close the transistor is to the critical path. These algorithms then try to scale as small as possible each transistor without violating its tolerable delay [36]. Another typical heuristic approach is to compute the slack at each gate in the circuit, where the slack of a gate corresponds to how much the gate can be slowed down without affecting the critical delay of the circuit. Sub-circuits with slacks greater than zero are processed, and the sizes of the transistors reduced until the slack becomes zero, or the transistors are all minimum size. Variants of the above approach are presented in [144] and [8].

Among these power optimization techniques, also transistor re-ordering deserves to be mentioned. As transistors arrangement affects the power consumption of a circuit, this method rearranges transistors to minimize their switching activity, thus the dynamic power [79] [143].

Dynamic and Pass-Transistor Logic

In CMOS static logic, voltage is always fed to the nets by a conducting path from the net to the supply rails. In the opposite, *Dynamic Logic* [43] nets can pass through the operating condition in which there is no path to the rails, and voltages are fed through the charge stored on nets capacitances. For this reason, the clock period is divided into two phases: *pre-charge* and *evaluation* phase. During pre-charge, the output is charged to Vdd. Then, during the next clock phase, the logic function is evaluated by an NMOS network discharging the output node, if necessary. Historically, dynamic design styles have shown relevant low-power properties, as an example, by having reduced device counts. In practice, dynamic circuits have several disadvantages. For instance, each of the pre-charge transistors in the chip must be driven by a clock signal. This implies a dense clock distribution network and its associated capacitance and driving circuitry.

These components can contribute significantly to the chip power. Besides, having every single gate driven by the clock, skew issues become even more critical and challenging to handle.

As for dynamic logic, *Pass-transistor Logic* [162] reduces the transistor counts, lowering circuit load capacitance. In the past, this made pass-transistor logic attractive as a low-power circuit approach. However, like dynamic logic, pass-transistor circuits suffer from several drawbacks. Pass transistors have asymmetrical voltage driving capabilities, and the voltage gap between high and low logic levels decreases at each stage. In summary, there might be conditions in which pass-transistor logic is more power efficient than fully-CMOS logic; however, from a general point of view, the benefits of this technique are limited if compared to the savings obtained by higher level approaches.

2.3 Dynamic Power Optimization Schemes

Energy efficiency is one of the major design concerns for today's digital Integrated Circuits (ICs). *Dynamic power management* is a design methodology that allows ICs to dynamically re-configure the Energy/Performance tradeoff according to the input workload (e.g., user's requests) and the external environment conditions. Several embodiments of this scheme have been proposed in the literature. Next sessions disclose their implementation and efficiency.

2.3.1 Dynamic Voltage Scaling (DVS)

Dynamic voltage scaling (DVS) is an effective approach for energy reduction of integrated circuits. DVS was originally conceived for low-power processor designs [164] and the main idea behind this method is straightforward: during stages of low processor utilization, the performance level can be reduced such that tasks can be completed "just in time." As the processor frequency is cut down also the supply voltage can be scaled. As shown by the equations 2.6, the frequency reduction combined with a quadratic reduction of the supply voltage results in an approximately cubic drop of power consumption. However, with reduced frequency the time to complete a task increases, leading to an overall quadratic reduction in the energy to complete a task.

$$Power \propto Vdd^2 \cdot F_{clk} \qquad Energy \propto Vdd^2 \tag{2.6}$$

DVS is, therefore, an effective strategy to reduce the energy consumption of a processor, especially under wide variations in workload as it is common in mobile applications. Extensive work has been performed on the methodology to determine the voltage scaling policies that maximize the energy savings [41] [40]. In a real system, Operating System (OS) is in charge to set frequency and voltage dynamically in order to match the changing demands for processing power. Hence, DVS requires algorithms, called *voltage schedulers*, to determine the processor operating frequency at run-time. Several

works in the literature evaluate scheduling algorithms in a verification environment which consist of an energy-accurate cycle-level simulator [120].

Furthermore, as shown in [164], the operating voltage range of processors can be significantly extended through additional design effort. Indeed, a lower limit of the voltage range can be determined for optimal energy efficiency. The optimal voltage limit depends on two factors: the power/delay trade-offs at low operating voltages and the workload characteristics of the specific processor.

2.3.2 Dynamic Voltage Frequency Scaling (DVFS)

The continuous progress in microprocessors has been supported mostly by technology scaling, which has been resulted in exponential growth of both device density and performance. This trend opened the way to the revolution of the digital System-on-Chips (SoCs) [157]. It is well known that power consumption is the most stringent constraint for the growth of SoCs [157]. However, as the technology scaling entered nanometer regime (100 nm), supply-voltage drop led CMOS devices to face many problems such as increasing leakage currents, large parameter variations, low reliability, and yield [71]. This represented a bottleneck for clock frequency increase. Therefore, in order to ensure computing performance growth without breaching power dissipation requirements, high- performance microprocessors moved to multi-core/many-core paradigm [45] [117]. As explained in 2.2.1, *Parallelism* is one of the best ways to address the issue of power reduction while maintaining higher computation throughput with lower voltage and frequency. The result is a larger silicon area, but overall lower power dissipation and power density. This is one of the ideas behind the transition to multi-core/many-core. In addition, when high-performance systems need to meet a low energy budget, the availability of multiple processing units that can be turned-ON/OFF, or just slowed down depending on the actual workload, represents an efficient solution. Within this context, Dynamic Voltage Frequency Scaling (DVFS) has been proven to be the most effective technique to get close to minimum energy consumption. DVFS is based on a straightforward working principle, that is, reduce the supply voltage (Vdd) down to the minimum threshold that satisfies the frequency constraint (F_{clk}) imposed by the actual workload.

Originally applied to "monolithic" SoCs [110] [153], the degree of freedom made available by multi-processor SoCs (MP-SoCs) architectures enabled a more efficient core-based, i.e., fine-grained, DVFS implementation [74]. With a fine-grain DVFS, each core can be set working at a different operating point in the $[F_{clk}, Vdd]$ space; this allows to run multiple tasks asynchronously and bring down the minimum-energy point of the whole SoC. An example of fine-grain DVFS on massively parallel platforms is given in [149], where 167 processors are orchestrated over a wide frequency range achieving minimum power consumptions, from 1.07 GHz - 47.5 mW at 1.2 V to 66 MHz - 608μW at 0.675 V. As an add-on feature, fine-grain DVFS is a perfect knob to compensate and/or mitigate variations due to Process, Voltage and Temperature (PVT) fluctuations

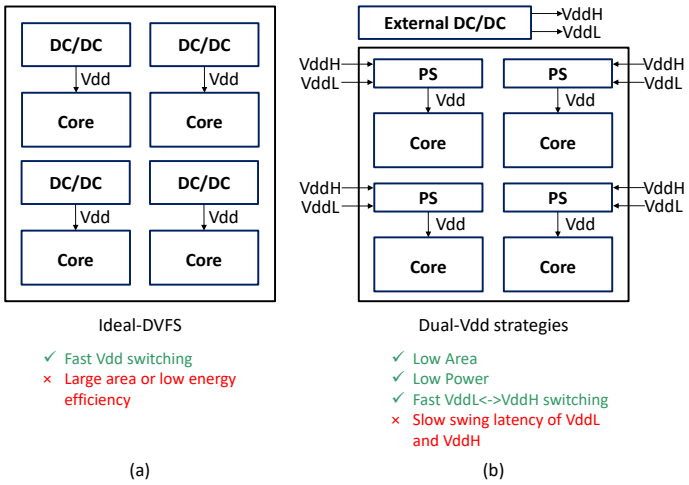


Figure 2.4: Comparison between ideal-DVFS and dual-Vdd strategies [118].

that affect different cores after fabrication and during the lifetime of the circuit [33].

2.3.3 Beyond the DVFS: Fine-Grain Vdd-Hopping

A practical use of DVFS on MP-SoCs deals with the availability of programmable on-chip Vdd regulators that can deliver the supply voltage with fine resolution step and fast swing (Fig. 2.4a). Unfortunately, the use of integrated DC/DC converters is made impractical due to high implementation costs. Indeed, on-chip DC/DC converters fabricated with today's technologies may occupy a considerable silicon area due to the low integration density of the components they contain, e.g., capacitors and inductors [149]. The picture gets even more complicated if one considers that every single core should be equipped with a dedicated converter.

The challenge faced by several works in literature is to achieve, or at least get close to, the efficiency of high-resolution DVFS, ideal-DVFS hereafter, with a discrete set of supply voltages. In their more general embodiment, *discrete-DVFS* strategies use two Vdd levels ($VddL$ and $VddH$) generated off-chip through external voltage regulators and evenly distributed across the die (Fig. 2.4b). The absolute values of $VddL$ and $VddH$ are shifted up/down depending on the workload, while each core is fed with the proper Vdd employing dedicated power switches (PS). Even though this design option offers a practical solution with a low impact on area and power, it comes with a speed penalty due to high voltage swing latency of the external voltage regulators [115]. Nevertheless, this is an acceptable cost as the voltage scaling process typically applies at a low rate.

The two most representative cases of discrete-DVFS are the Vdd-Hopping [98] and the Vdd-Dithering [9]. The Vdd-Hopping is a basic scheme in which the supply voltage range is split into a discrete set of values, two or more depending on the external voltage regulator (Figure 2.5a); the proper Vdd is selected among the available values

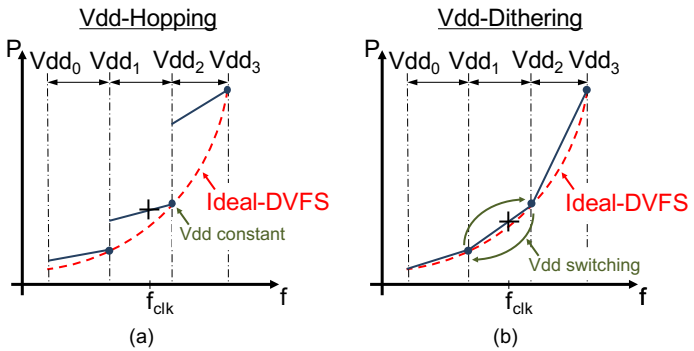


Figure 2.5: Frequency-Power tradeoff of existing DVFS strategies [118].

such that the frequency constraint is met. The Vdd-Dithering scheme is a more elaborated, yet precise scheme that implements a Vdd time-sharing strategy. Differently from Vdd-Hopping, the Vdd is made switching from low ($VddL$) to high ($VddH$), leading the core to an average frequency equals to the frequency constraint (Figure 2.5b). The power-frequency tradeoff obtained through Vdd-Dithering can be seen as a linear approximation of ideal-DVFS

While the techniques mentioned above aim at pushing power consumption close to ideal-DVFS while using a discrete set of supply voltages, *Ultra-Fine Grain Vdd-Hopping* (FINE-VH) is a practical methodology that brings DVFS beyond its theoretical limit [119], [118].

FINE-VH leverages the working principle of Vdd-Hopping applied *within-the-core* through an ultra-fine dynamic dual-Vdd. As reported in Figure 2.6, FINE-VH is a tile-based strategy, i.e., it applies to regular sections of the layout.

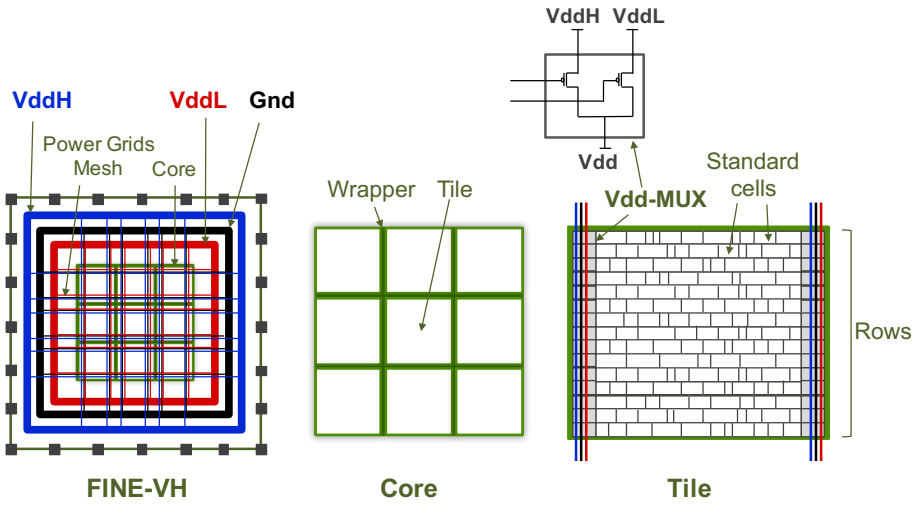


Figure 2.6: FINE-VH within-the-core layout partitioning and tiles organization [118].

Working with multiple voltages within the same functional unit raises several concerns during the place&route stages, e.g., area overhead and timing closure, due to layout fragmentation and standard cell displacement. Moreover, the static power consumption increases due to the leakage currents of those logic gates driven by tiles powered at different Vdd. An optimal body-bias assignment technique can compensate for the intra-tile leakage at no delay penalty.

2.4 Adaptive Power Management (APM)

2.4.1 Better-Than-Worst-Case Design

As extensively explained in previous sections, the advent on nanometer feature sizes in silicon fabrication has triggered three main challenges for ICs design: *(i)* the design complexity having a larger transistor budgets, *(ii)* the design uncertainty due to increasing environmental and process variation, *(iii)* the protection from faults, especially soft errors. Limited energy budgets and increasing time-to-market pressure have raised even more concerns about these challenges.

Better Than Worst-Case design (BTWC) faced these design demands through a novel strategy that brought a huge innovation in ICs design style [5]. BTWC proposed to decouple the constraints of correctness and robustness from those of performance and power. The approach splits designs into two primary components: a core design component and a compliance checker. The core design component is responsible for performance and power efficient computing; the checker is in charge of verifying the correctness of the core computation. In this way, the core design is relieved from the overall concerns of operating compliance that are transferred to the checker component. With more relaxed compliance constraints in the core, design challenges can be more effectively addressed.

In addition, BTWC core/checker paradigm enabled adaptive power management for energy-efficient ICs design. Indeed, BTWC has been proved to be more efficient than the classical low-power techniques, discussed in Section 2.3, whose weakness lies in their intrinsic "always-correct" nature which dictates a theoretic lower bound of the energy savings. Let us take a straightforward implementation of voltage scaling for instance. There is a minimum supply voltage Vdd_{min}, imposed by the worst-case condition analysis, at which timing paths violate the set-up time; below such threshold, timing faults do arise, and logic errors propagate. A further reduction of the Vdd can be only accomplished with modification of the circuit, i.e., as BTWC paradigm proposes, through a checker block that monitors the system compliance.

BTWC design idea imposes a constraint onto the checker component, i.e., it must be simple to implement to avoid system complexity overhead. According to the literature [5], mainly circuit-timing checkers match this constraint; such checker components validate the timing compliance of circuit-level computations. Using this capability to

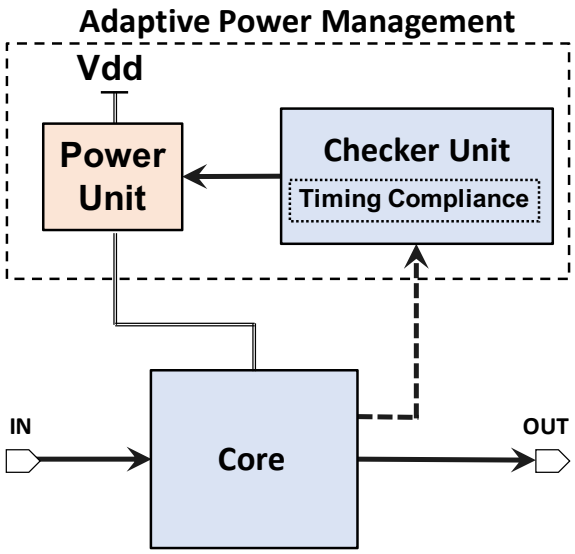


Figure 2.7: BTWC paradigm abstract view and Adaptive Power Management.

monitor timing errors, BTWC design can eliminate power-hungry voltage margins. Indeed, the checker mechanism can be used to adaptively manage the power delivery by adjusting the supply voltage according to the timing compliance state of the core circuit, as shown in Figure 2.7. Finally, if BTWC is combined with a dedicated fault recovery mechanisms the design can be pushed beyond the theoretical limit of classical low-power techniques yet ensuring an "always-correct" computation.

An in-depth analysis of adaptive low-power management strategies, in particular *Adaptive Voltage Scaling*, and the implementation details of BTWC designs based on timing-compliance checker mechanisms are reported in the following sections.

2.4.2 Adaptive Voltage Scaling (AVS)

Among the low-power techniques proposed in the literature, *Adaptive Voltage Scaling* (AVS) [37] [32] [134] represents the very first example of speculative power management approach which perfectly match with the BTWC paradigm. The key idea of AVS is that of tuning the supply voltage by monitoring the error rate coming from checker mechanism. In other words, AVS technique gives circuits the capability to understand, at run-time, depending on the input workload, i.e., the flow of data, if/when there is margin to further reduce the energy consumptions below the theoretical minimum imposed by worst-case operating condition.

As explained, the "adaptive" nature of AVS power management comes from the fact that the optimal voltage configuration is identified through the measurement of the input workload, which is, in turn, an indirect measurement of the circuit dynamic paths distribution made by the checker component. This represents the key difference between AVS and DVS. Indeed, DVS is an open-loop system as it employs a voltage/task

one-to-one mapping pre-defined at design time through a voltage/workload characterization at worst-case conditions. Thus, in order to guarantee robust operations under worst-case, significant energy scaling margin is left. In contrast, AVS can adaptively (i.e., according to the context) reduces the worst-case voltage margins improving the overall energy efficiency of the design.

In summary, the crucial difference between "dynamic" and "adaptive" power management lies in the nature of the information used for setting the target voltage while an application is run. DVS, as an open-loop strategy, schedule (or predict) the operating voltage according to a design-time (or static) workload characterization; AVS employs a feedback mechanism that keeps probing the actual on-chip conditions, specifically, the timing compliance, and according to that set the target voltage. As a closed-loop power management approach, AVS requires a stable and safe method to detect the timing faults. Several of fault detection mechanisms are available in the literature [123] and are discussed in the next sections.

2.5 Implementing APM: Timing Speculation for AVS

The previous section has highlighted how BTWC core/checker design paradigm enables adaptive power management for energy-efficient digital systems. As the most effective checker components are designed to be sensitive to the circuit timing compliance, *Timing Speculation* principle has been exploited to implement adaptive low-power techniques. This strategy relies upon the probabilistic assumption that only a small and infrequent sub-set of input patterns sensitize the longest circuit timing paths; for Vdd below Vdd_{min} those paths are rarely activated and, therefore, timing faults remain latent. In the event they get excited, latent faults become true faults but the resulting error can be recovered through some correction mechanism, guaranteeing an "always-correct" computation. As a result, the circuit operates at minimum energy consumption point for most of the time.

As the key idea of AVS is to adaptively tune the supply voltage by monitoring the timing faults rate measured by an error detection mechanism, it comes naturally to consider AVS as the power management that best exploits the timing speculation principle. Several embodiments of AVS are available in the literature [123]; they differ in terms of *(i)* the method used to detect faults and/or *(ii)* the circuit optimization applied to recover the occurrence of errors. Figure 2.8 reports a classification of AVS implementation techniques grouped according to the timing faults detection strategy.

In-situ timing Monitors (or Sensors) provide the power manager with real-time and local feedback on the health of the chip. As they require measurement within the internal structure of a circuit, monitors typically use double-sensing Flip-Flop on the most critical paths of the circuits. Depending on the implementation, a timing sensor can *(a)* detect an occurred error and correct it through a recovery mechanism; *(b)* predict the

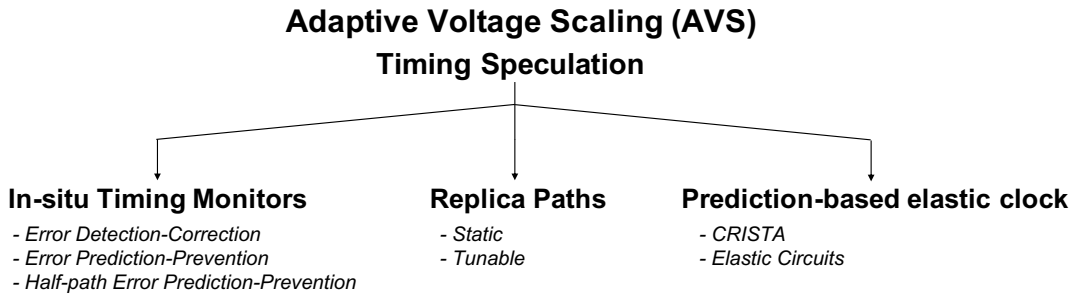


Figure 2.8: Adaptive Voltage Scaling driven by Timing Speculation: a classification.

occurrence of an error by measuring the slack of activated path; *(c)* predict the occurrence of an error by measuring a delay at the half point of a critical path. By predicting a possible timing fault, both *(b)* and *(c)* prevent the occurrence and, thus, the propagation of the error. In-situ sensing ensures small area overhead and fine-grained observability of the timing errors.

On the contrary, *Replica Paths* placed on the border of the circuit are in charge of measuring the effect of the voltage scaling preventing the potential occurrence of timing faults in the functional paths of the circuit. This technique is less intrusive on-chip but the cost to pay is a more coarse-grained capability of the timing faults detection.

Prediction-based elastic Clock techniques operate at synthesis level isolating the critical (long) paths of the design and provide an alert mechanism raised when they are activated. It ensures a fine-grained capability of detecting timing errors at the cost of a full modification of the original circuit due to onerous re-synthesis stages.

For the sake of clarity, all of these AVS techniques perform an efficient energy scaling yet ensuring "alway-correct" computations. A more detailed overview of AVS methods is reported in the following sections.

2.5.1 In-situ Timing Monitors: Error Detection-Correction

This class of AVS approaches allows the faults to occur by operating at the edge of failure, i.e., reducing the supply voltage to consume the conservative slack guard-band introduced by the worst-case design. To manage the timing errors, these approaches require two main mechanisms: *(i)* an error detection system, i.e., a mechanism to detect the incorrect state values caused by the timing errors; *(ii)* an error correction mechanism triggered upon an error detection flag to compensate the effects of errors during system operation.

Razor

Within this class, Razor [39, 38] is the primary reference. As reported in Figure 2.9, it is based on Razor-FFs that check the correctness of the circuit through a time-skewed comparison of the sampled values in the main Flip-Flop (FF) and a Shadow

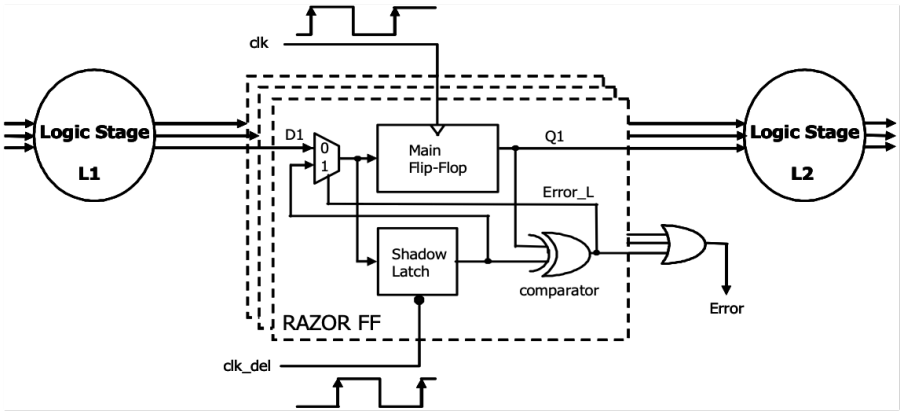


Figure 2.9: Abstract view of Razor FF and augmented pipeline [38].

latch. As previously highlighted, the key idea of Razor-based AVS is that of tuning the supply voltage by monitoring the error rate. This reduces the voltage margins improving energy efficiency. Razor finds application in microprocessors architectures, where a three-stage mechanism implements the error recovery: first, as the timing error occur, the processor pipeline is stalled; second, FFs in violation are refreshed with the correct value sampled in the shadow latch (2.9); third, the last pipe-cycle is re-executed. Both error detection and correction are performed locally, i.e., on the Razor-FFs.

As Razor-FF plays with a time-skewed clock signal, setup and hold constraints at the main FF input might not be respected leading the state of the FF to become metastable. A metastable signal increases critical path delay which can cause a shadow latch in the succeeding pipeline stage to capture erroneous data, thereby leading to incorrect execution. Also, a metastable FF output can be inconsistently interpreted by the error comparator and the downstream logic. Hence, an additional detector has been introduced to correctly flag the occurrence of metastability at the output of the main FF [30]. The outputs of the metastability detector and the error comparator are ORed to generate the signal of the Razor-FF. Thus, the system reacts to the occurrence of metastability exactly in the same way as it reacts to a conventional timing failure.

Energy savings due to Razor-based AVS on a 64-bit processor are 50% over worst-case operating conditions; this result is obtained by scaling supply voltage to achieve a 0.1% targeted error rate, at a fixed frequency of 120 MHz [30].

Razor II

As a further effort to avoid possible metastability issues, Razor II [31] has been proposed as an extension of Razor. This solution fights metastability by checking the occurrence of errors through a transition detector instead of using a shadow latch (Figure 2.10). Moreover, it shifts the error-recovery mechanism at a “architecture” level, that is, the FFs are in charge of error detection, while the correction is performed through a

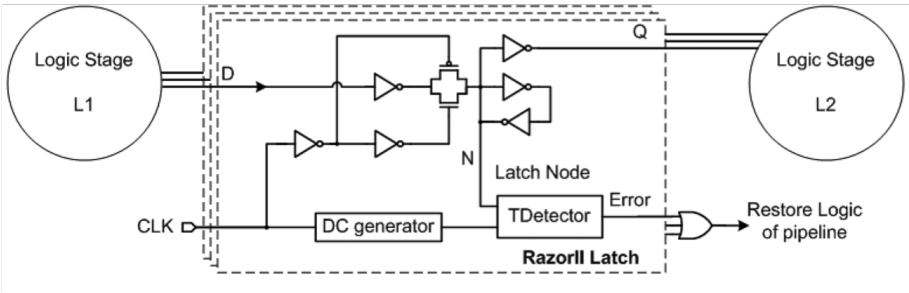


Figure 2.10: Razor II implementation [31].

full replay of the latest instruction. Instruction replay is achieved by check-pointing the Program Counter (PC) register. The PC register is scrolled along the processor pipeline. When an error is detected, the entire pipeline is flushed, and the PC in the fetch stage is overwritten with the check-pointed PC, i.e., the PC of the instruction has generated the timing error. Normal instruction execution resumes from then on. Since an erroneous instruction is re-executed through the pipeline during replay, the same instruction can suffer repeated timing errors. Hence, the clock frequency is halved to allow guaranteed completion.

Razor II strategy significantly reduces the size the Razor-FF at the cost of overall performance penalties. The results collected on a 64-bit microprocessor show 33% average energy savings and an instruction-per-cycle degradation of 0.2%.

Error-Detection Sequential (EDS)

Authors in [14, 13] proposed an alternative circuit for error detection and correction which prevent metastability by replacing the Razor FFs with time borrowing Latches. They called this technique *Error-Detection Sequential* (EDS). An AVS-based design can be implemented by replacing FFs on the critical paths with EDS circuits. As shown in Figure 2.11, the EDS is a double-sampling circuit augmented with a time-borrowing latch mechanism. It consists of the main FF, a shadow latch and an XOR gate. Similarly to Razor, the main FF and shadow latch sample the data on the critical path end-point at the rising and falling clock edges, respectively. The XOR logic gate compares these time-skewed samples and in case of mismatch generates an *ERROR* signal. Specifically, if the main FF input does not meet the setup time constraint due to voltage scaling or variations, the latch and main FF outputs differ, resulting in a *ERROR* signal rising. The generated *ERROR* flag can be propagated to the rest of the chip to invalidate the erroneously executed operation and trigger a proper recovery mechanism for correction, which is architectural as in the Razor II case. Silicon measurements on microprocessor circuit show an efficient AVS profile with remarkable energy reduction (37%). However, the most significant limitation of this technique concerns the need to the to convert the standard design into a latch-based design.

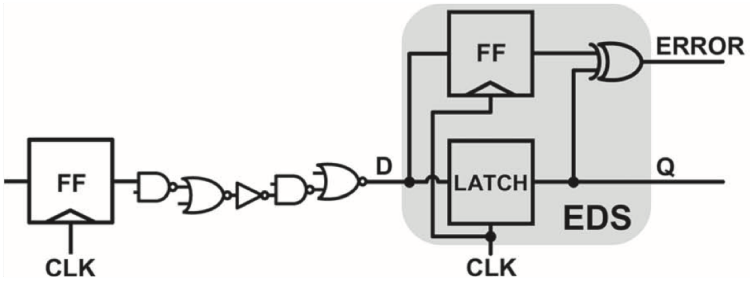


Figure 2.11: Abstract view of EDS [14].

Synthesis-level Slack Redistribution

Recent works introduced circuit/architectural level optimization for a more efficient AVS based on error detection-correction [61][64] [51]. These techniques proposed to reduce the minimum allowed Vdd by means of a timing slack redistribution. With such techniques the most frequently exercised paths are made faster, thereby forcing timing errors on the most infrequent paths [64]. This choice maximizes the Vdd lowering range while reducing the power consumption under a given constraint of performance loss. Experiments upon an open source processor demonstrate 23% power savings with an error rate of 1% and an area overhead of 2.7% [64].

Reshaping the original path distribution is a (static) design approach which may result too weak (or too pessimistic) as most critical paths may change due to process variations and/or time-dependent variations [18, 155].

On the limitation of standard error detection-correction schemes

Existing Error Detection-Correction strategies suffer from the so-called *short-path race*, i.e., the overlapping between short- and long-paths during error detection, which imposes the adoption of tedious hold-fixing procedures run at design time. Hold-fixing (when converging) may induce a substantial modification of the circuit's characteristic. Firstly, depending on the characteristics of the circuit, hold-fixing may lead to significant area/power overhead. As reported in [70], for a simple test-case made up of three 16-bits multipliers, the area grows by 2.1x, while energy per operation increases up to 81% w.r.t. the baseline circuit. Furthermore, paths are substantially shifted towards the clock edge. This issue (as proven by our simulation results) represents a severe impediment for effective use of AVS [126]..

2.5.2 In-situ Timing Monitors: Error Prediction-Prevention

Unlike Error Detection-Correction techniques, this class of circuits addresses timing speculation in a complementary way. The timing error is predicted by tracking the

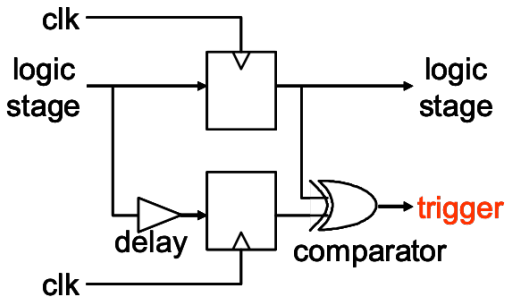


Figure 2.12: Abstract view of a Canary FF [131].

slack of the activated paths and prevented by raising the supply voltage. The most representative device of this class is the Canary FF [131]. As reported in Figure 2.12, Canary FF circuit uses the same double-sampling mechanism of Razor, but it feeds a delayed input signal to the shadow FF (rather than a delayed clock). The delay amount is called *prediction window* and represents the timing slack to which the Canary FF is sensitive. Specifically, a comparator is in charge of checking whether a data transition occurs in the prediction window, meaning that a path with a slack lower or equal to prediction window has been activated, and thus, raising an error trigger (or warning). As reported in the literature, the AVS is driven by a warning rate which measures the frequency of the potential errors during a monitoring period.

This technique does not introduce the short-path races issue with the tedious and expensive hold-time constraint fixing procedure, needed in Razor technique. Thus, less area overhead and easier timing closure can be achieved. Moreover, Canary-based AVS does not require the synthesis of a delayed clock tree.

However, Canary FFs can only predict and prevent potential errors from propagating as the supply voltage is scaled. If a critical path activation occurs in the current clock cycle, the main FF stores an erroneous sample, thus the timing error cannot be detected nor prevented, and the fault may propagate in the fan-out logic stage. In other words, Canary can only prevent future errors, and cannot recognize errors that occurred in the current clock cycle. Due to this reason, Canary has been mostly applied in error-resilient applications, where a certain amount of errors can be tolerated (e.g., image/video codec, base-band processor) [131]. Very conservative configurations of the prediction window can bring canary-based AVS to quasi always-correct computation [44] with error occurrence probability very close to zero. However, recent works like [10] [130], have shown that "always-correct" computation can be obtained in specific applications (e.g., BCH encoder/decoder) by setting for each canary FF a prediction window width that follows the slack of the observed critical path.

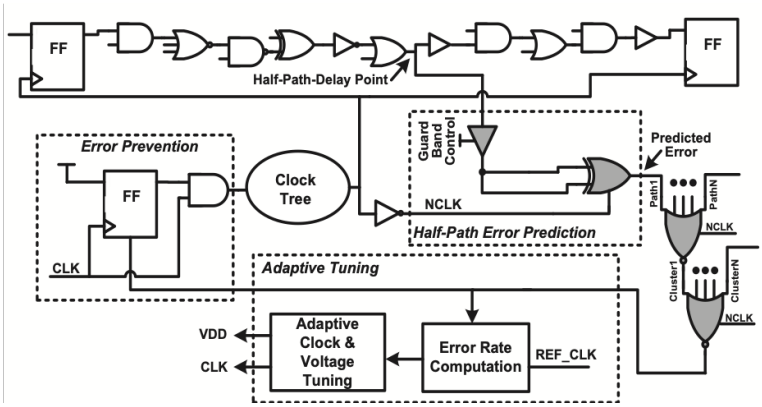


Figure 2.13: Abstract view of AVS based on Half-Path Error Prediction-Prevention [165].

2.5.3 In-situ Timing Monitors: Half-path Error Prediction-Prevention

The core idea of this class of techniques for AVS is that the timing error can be predicted by detecting the late-arriving data transitions at an intermediate node in the critical path instead of at the end-point FF. Then, the observed potential timing error can be prevented through dynamic clock gating, as reported in Figure 2.13. In [165] [135] [107] the half-path-delay point is marked as the optimal place for detecting a late-arriving transitions. Any transition occurring after the falling edge of the clock at the half-path-delay point can be identified as a potential timing error.

Unlike Error Detection-Correction techniques, this method does not need to fix the short-path race issue as a late transition is observed at critical path half-point. This reduces the area/power overhead and makes the timing closure much easier especially for ultra-low-voltage operation. Also, the error prediction is oriented toward the faults in the current clock cycle, rather than toward future errors like in the Canary FF technique. Furthermore, error prevention is performed by dynamic clock gating, which can be easily integrated into general digital designs. Practical use of this kind of AVS technique is reported in [90] where an energy-efficient sensor node processor is presented for intelligent sensing in the Internet of Things applications.

A more recent work, SlackProbe [82], has shown that in-situ monitors can be placed at different intermediate nets along the circuit paths. This monitors track the slack of critical paths and raise a warning when potential timing violations may occur.

The main drawbacks of these techniques can be summarized as follow: *(i)* possible timing error miss predictions can happen if an error raises in the second half of the critical paths or after the nets where the in-situ sensors have been placed leading to functional failure. *(ii)* False alarm detection can occur when an error warning raised in the first half of the critical path is then compensated in the second-half; this condition might lead to severe performance overhead due to unnecessary error prevention mechanism activations (i.e., clock gating). *(iii)* Design complexity and area overhead due to

the sensors placement in the case of a large number of critical paths; also, the complexity to identify the right half- or intermediate-points along critical paths in the presence of variations.

2.5.4 Replica-Paths Error Prevention

Unlike in-situ timing monitors, *Replica-paths (or external sensors monitors)* are placed on the same chip die, but outside of the functional paths enabling less intrusive timing monitoring operations. This technique relies upon the principle for which critical paths replica of a circuit can be used to measure the effect of the voltage scaling preventing timing faults from occurring in the functional paths of the circuit [78] [150]. Figure 2.14 shows the AVS circuit is composed by three main parts: *(i)* the voltage scaled circuit, *(ii)* the voltage control logic, and *(iii)* the critical paths replica. According to the literature, critical paths replica are often implemented through three components: replicated logic, delay buffers, and an edge detector. The latter checks the transition edge timing at the end-point of the critical path replica for every clock cycle and generates a warning signal when the edge is late. Replica circuits should include many paths of the main circuit as to have the most similar sensitivity to variations during voltage scaling. Obviously, this comes with a large area cost. However, compact replica paths have been proposed in the literature to overcome this issue. They can be implemented through a design-time sensitivity characterization [69] or by integrating tunable replica paths that can be set at post-fabrication time [151].

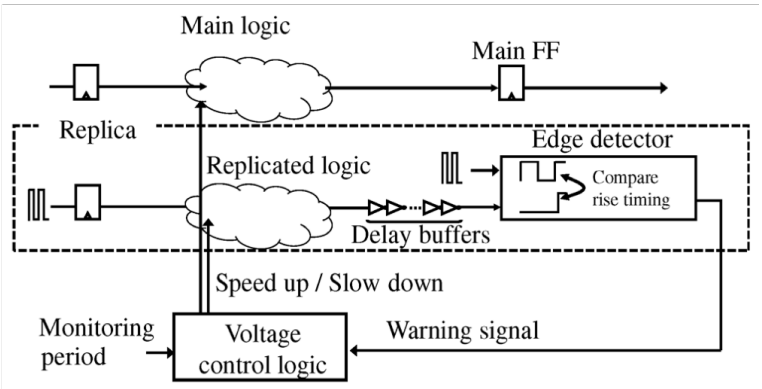


Figure 2.14: Abstract view of Replica-paths based AVS [96].

Among on-chip sensing and adaptive tuning circuits, the critical path replica technique has the lowest overhead, but it cannot capture the local variations due to the location difference between the actual critical path and the replica circuit. Also, unlike in-situ timing monitors, replica paths are not sensitive to the input workload, making AVS very conservative and not responsive to the context (i.e., the flow of the input data).

2.5.5 Prediction based elastic-clocking

The most representative technique among this class of AVS powered by timing speculation is proposed in [50]. The authors introduce a design paradigm, called CRISTA, that implements AVS under a desired frequency constraint. The basic idea is to isolate the most critical paths through a custom re-synthesis stage that reshapes the original paths distribution. The Vdd is then tuned such that the most critical paths violate the set-up time, while the remaining non-critical paths run error-free. The activation of the critical paths is predicted by a dedicated control logic that works as a logic error sensor. The flag returned by this sensor triggers timing speculations: an extra clock-cycle is given when long-paths are excited. The CRISTA design methodology ensures activation rates of the long-paths are low enough to avoid excessive performance penalties. For a two-stage pipelined ALU, CRISTA allows reducing the power consumption by 40% with a mere 9% area overhead.

The solution presented in [99, 21] is a variant of the CRISTA paradigm which still exploits the concept of variable latency units. Results show 45% power savings w.r.t. the baseline implementation. When tested on a 5-stage pipeline micro-architecture using SPEC2K benchmarks the throughput penalty is 4%.

No matter its actual implementation, CRISTA is applied at design time, namely, both the selection of the critical paths and the synthesis of the activation function (or variable latency units) are done statically using some a-priori knowledge of the circuit. However, process variations represent a serious concern; the path distribution may change due to manufacturing imperfections, and some critical paths may result uncovered by the activation function (defined at design time). As far as reported in the literature, a practical solution to make CRISTA adaptive/tunable for post-silicon calibration does not exist; indeed, “guard-banding”, i.e., an over-selection of the critical paths that have to be isolated, seems the only option available. Unfortunately, this exacerbates the design overhead. Also, CRISTA requires additional logic synthesis stages which are not available in standard design kits.

Chapter 3

Energy-Accuracy Scaling: a New Paradigm

As already introduced in Chapter 1.1, the energy efficiency of Integrated Circuits (ICs) is a major bottleneck in a wide range of today's applications, from SoC for the Internet of Things (IoT) up to cloud and datacenter-scale computing (e.g., servers and GPUs). The last decades' energy reduction trend in ICs supported by technology scaling, and architectural-/circuit-level optimization strategies (e.g., deep parallelism and aggressive voltage scaling) has approached its physical limits. Among the recent works in the literature, an application-level approach, defined as *Energy-Accuracy Scaling* (EAS), proposes to embed the system accuracy as a new dimension into the design/optimization space to have a further knob for adjusting circuits energy-efficiency [1] [3].

In general, the *Accuracy* of a circuit/system represents the capability to deliver output results as conforming as possible to the expected result. Thus, an application that can admit inaccurate computation by tolerating the occurrence of errors is called error-resilient [16] [85] [89]. Real-world is plenty of systems that are resilient to errors in fields like multimedia, where a human end-user may not notice small degradation in images computed by circuits affected by errors [111], or artificial intelligence where algorithms have intrinsic mechanisms to deal with inaccurate or uncertain data (e.g., Neural Networks) [16]. These applications prove that output accuracy can be naturally traded for energy savings if the user does not perceive the quality degradation. A general rule for EAS is summarized in Figure 3.1: higher accuracy can be achieved by paying a higher energy per task, or more interestingly, the computation energy can be reduced if the running application can tolerate lower accuracy.

Following sections show that EAS is the key to pushing energy efficiency beyond the limits encountered with classic energy scaling strategies presented in Chapter 2, exploiting the error-resilience of specific applications. However, let us discuss firstly a taxonomy of functional- and circuit-level techniques that enables energy-accuracy scalable digital ICs.

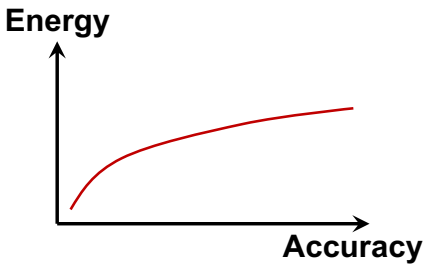


Figure 3.1: Energy-Accuracy tradeoff trend for error-resilient ICs.

3.1 Energy-Accuracy Scaling (EAS): a Taxonomy

This section introduces a revisited taxonomy of approaches that support EAS classified according to one main feature: *the capability of tuning the energy-accuracy tradeoff*. In general, as shown in Figure 3.2, three main categories can be identified for EAS strategy: *(i)* Static, *(ii)* Dynamic and *(iii)* Adaptive.

(i) In the *Static* approach, the energy-accuracy tradeoff is fixed at design-time (i.e., not tunable) by functional speculation rules, i.e., a modification of the circuit logic functionality through algorithmic simplifications and/or circuit approximations. These design transformations induce area/energy savings in exchange for an accuracy loss threshold. The output errors cannot be either mitigated nor recovered at run-time but they are guaranteed to be bounded by a worst-case magnitude. Some examples of techniques in this class are the circuits based on approximate computing and inferential logic.

(ii) Unlike static methods, *Dynamic* EAS enables a tunable energy-accuracy tradeoff at run-time. The collection of the energy-accuracy operating points are pre-defined at design-time through a circuit characterization at worst-case conditions. During the run-time stage, this EAS technique can schedule (or predict) the operating point according to the design-time (or statistical) characterization, ensuring a worst-case accuracy drop for energy savings. Dynamic Voltage Accuracy (Frequency) Scaling, a.k.a. DVAS (DVAFS), is an example of dynamic EAS indeed, as it allows voltage scaling through accuracy reduction without inducing timing faults, i.e., matching worst-case conditions.
Dynamic techniques are by nature open-loop systems as they do not receive any feedback information directly from the chip (e.g., input workload, output/timing compliance). However, environmental conditions can trigger operating point switch; for example, in a JPEG compression of a picture taken from a camera, a specific accuracy drop can be tolerated when the quality of lighting is poor, as sensed by a light sensor.

(iii) By definition, *Adaptive* EAS is a closed-loop scaling approach for which the optimal energy-accuracy tradeoff is achieved by measuring defined quality metrics

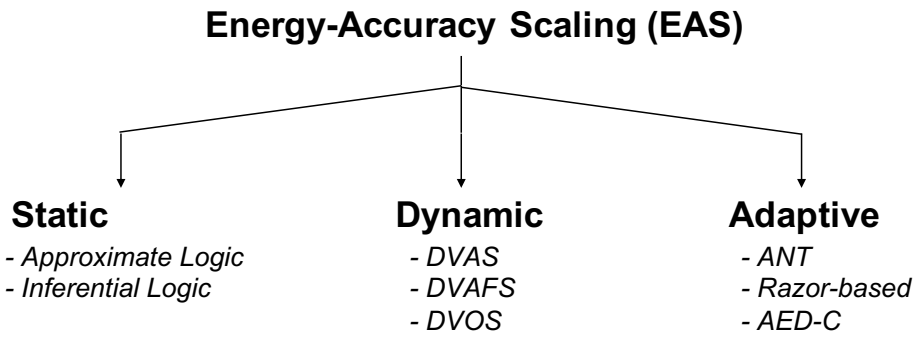


Figure 3.2: Energy-Accuracy Scaling techniques classification.

directly on-chip, at run-time. This metrics can be obtained by a direct measure the output accuracy, or by an indirect estimation of the accuracy value. While in the former case an upper bound on the error can be fixed, e.g., Algorithm Noise Tolerant (ANT) systems, in the latter one the accuracy loss is kept under control through statistical tools, e.g., the errors rate measured by Razor timing monitors. Most importantly, the feedback loop allows the adaptive approaches to achieve a finer tunability of the energy-accuracy tradeoff beneath the margins introduced by the conservative energy-accuracy operating points defined for the Dynamic EAS.

These classes of EAS are discussed more formally in the following sections; for the sake of space, only the most representative techniques for each class are reviewed in details (please, refer to Figure 3.2).

3.2 Static EAS: speculation at Functional-level

As introduces in the previous section, in *Static* EAS the energy-accuracy tradeoff is fixed at design-time by functional speculation, i.e., a modification of the logic functionality through algorithmic or circuit simplifications which induce energy savings for a worst-case accuracy loss. Compared to Dynamic solutions, Static EAS cannot change the energy-accuracy tradeoff, i.e., the worst-case error and power savings cannot be tuned at run-time. Also, it does not integrate any feedback loop from the circuit, e.g., workload info or timing/output compliance; this implies the output error cannot be either mitigated nor recovered at run-time as Adaptive EAS strategies can do.

Two major applications belong to this class of technique: Approximate Logic and Inferential Logic. The former relies upon the Approximate Computing principle, extensively investigated in literature in arithmetic circuits. It found application several error-resilient applications, particularly in multimedia (e.g., Audio/Video processing).

The latter exploits Machine learning theories to describe Boolean logic functions as inference rules. As a result, statistical inference algorithms are leveraged to design combinational logic circuits, i.e., inferential circuits, that mimic boolean functions to a certain degree of accuracy.

The following sections briefly report some Approximate Logic Circuits design rules, then show how Inferential Logic is conceived for low-power circuits that run *quasi-exact* computation in error-resilient applications.

3.2.1 Approximate Logic

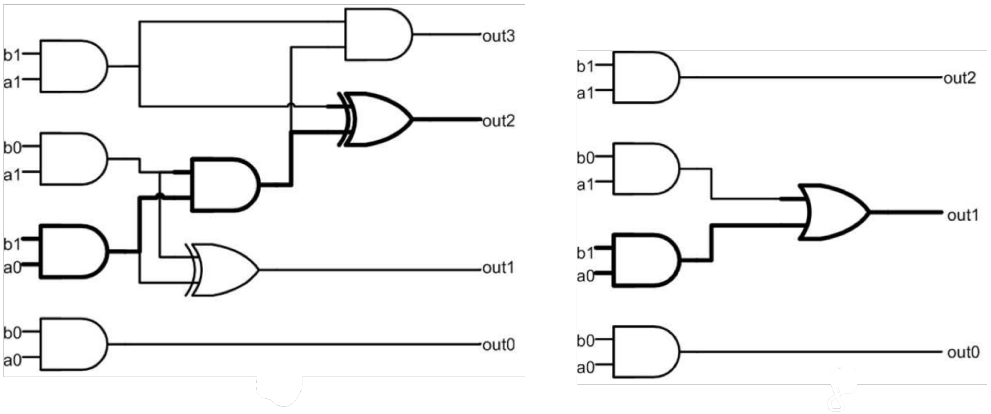
Functional-level approximation has been extensively used to reduce energy-accuracy through algorithmic and circuit simplifications. The rich literature on the approximate computing theme covers a wide scope of approximate adders [92] [48] [167] [166] [63] [97] [72] [4] and multipliers with truncated carry chain [94] [81] [77] [88] [106] [6] [132] which respectively lead to an energy reduction of 1.5-3.7× and 1.5-3.2×. Similarly, approximate DSP blocks have been investigated, such as FIR filters [141], Multiply-Accumulate units [27] [68], arithmetic processors [133], and Discrete Cosine Transform accelerators [116], with energy savings in 2× - 4.1× range.

Also, several frameworks proposed an automatic synthesis of approximate functions with power-quality target fixed at design-time. Some of these frameworks introduce an approximate replica of the circuit to check for timing faults of the main circuit [28], the extraction of those minterms which generate the minimum error rate [138], the statistical pruning of Boolean and arithmetic functions [86], and more general approaches for sequential [124] and combinational circuits [159]. The energy savings achieved by employing these frameworks ranges from 1.2 to 7.5×.

Energy can also be lowered by reducing the execution time required by tasks. For example, the execution time can be reduced by order reduction in digital filters [93], and sub-sampling of input data [27]. More in general, energy benefits can be obtained through the early termination of iterative algorithms ("anytime" algorithms) [95] and loop break [139]); monotonic output-accuracy increase over execution time can be guaranteed. An explanatory example among the galaxy of approximated multipliers can be used to clarify logic approximation techniques: a multiplier approximation through function under-design. Let us discuss it briefly.

Multiplier Approximation through Function Under-design

The authors of [77] propose to introduce error into a multiplier by manipulating its logic function, specifically by removing some Karnaugh Map entries. They designed an architecture that leverages a modified inaccurate 2×2 multiplier building block to implement larger multipliers. The structure of the resulting under-designed 2×2 multiplier is reported in Figure 3.3a, opposed to the exact function in Figure 3.3b. The inaccurate circuit is smaller and faster as the critical path delay is reduced. Also, fewer wires are



(a) (b)

Figure 3.3: Accurate (a) and Inaccurate (b) 2×2 multipliers with the critical paths [77].

required. Hence, such design is able to trade energy for accuracy by introducing in the design logic function approximation..

Larger inaccurate multipliers can be built upon 2×2 multiplier building blocks exploiting the rules of computer arithmetics for binary numbers (Figure 3.4). Such multipliers achieve an average power saving which ranges from 31.78% to 45.4% over corresponding accurate multiplier design, for an average error of 1.39% and 3.32% respectively. Using image filtering and JPEG compression applications, the inaccurate architecture can achieve 2× - 8× better Signal-Noise-Ratio (SNR) for the same power savings when compared to probabilistic based power-error tradeoff methods citegeorge2006probabilistic.

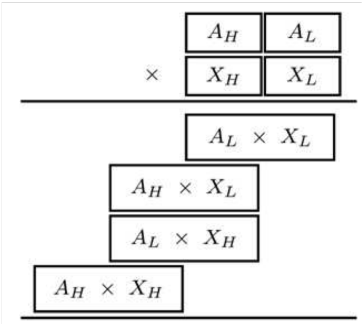


Figure 3.4: Building larger multipliers from smaller blocks.

3.2.2 Inferential Logic

In the Authors' view of [146], Machine Learning (ML) techniques should be exploited to implement logic circuits that mimic how the human brain works, i.e., like a

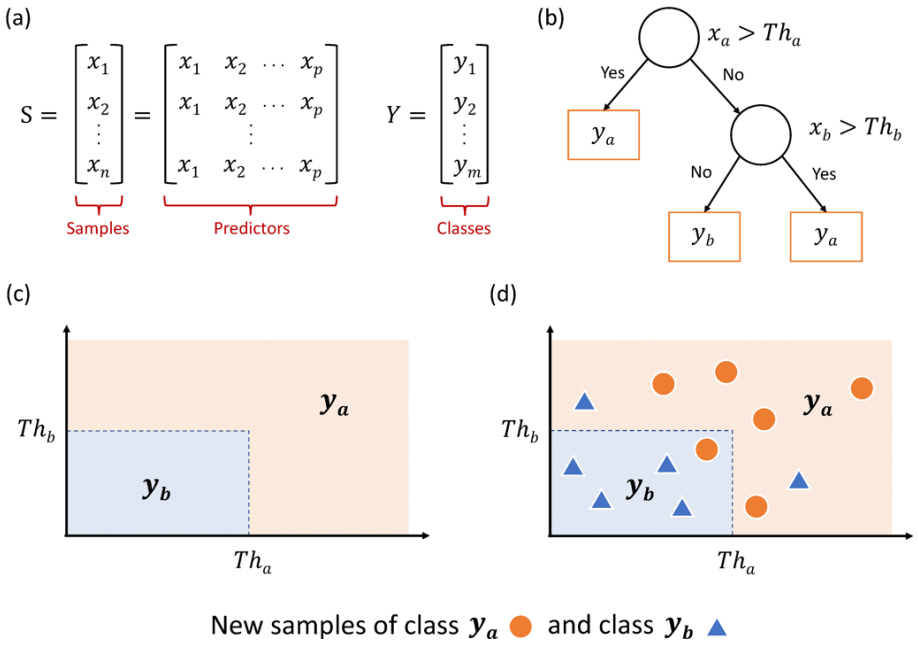


Figure 3.5: Classification problem: (a) analytical model, (b) abstract model representation using a Classification Tree, (c) input space partitioning, (d) new samples classification [147].

statistical inference engine [24] [148]. This paradigm shift encompasses the replacement of exact logic rules and Boolean operators in favor of statistical models and inference mechanisms, namely, to design inaccurate, yet energy-efficient logic functions [145] [147]. In this new design methodology, the representation of a generic Boolean function is built through a learning problem. The result is a logic function whose behavior is described by a more compact abstract model that serves as a statistical representation of the function itself. Such a representation is then used to infer the results of the Boolean function with a certain level of accuracy. Once mapped on a piece of hardware, the resulting circuit runs the quasi-exact computation of the logic function trading accuracy for energy savings; this logic block is called *Inferential Logic Circuit*. Let us focus on an explicative example of this approach: the implementation of quasi-exact logic function through Classification Trees, a very efficient and compact ML algorithm.

Training Classification Trees

Classification Trees (CTs) are machine learning tools belonging to supervised learning algorithms, used when prior knowledge of the problem is available, i.e., an observation set of samples properly labeled. CTs allow the identification of the most significant key features among observation samples. More formally, is it possible to split a generic classification problem in two main phases: *(i) Training*, during which a training data set, consisting of n samples labeled with one of the m available classes $y_i \in Y = \{y_1, \dots, y_m\}$

and described through p predictor variables $X = \{x_1, \dots, x_p\}$, as in Figure 3.5a, is used to learn an efficient abstract model representation, as the one reported in Figure 3.5b; *(ii) Validation*, during which a data set, made up of a new set of samples labeled with Y and described through X, is used to quantify the accuracy of the trained data model representation, as shown in Figure 3.5c which describes the input space partitioning due to the rules imposed by the model, and Figure 3.5d that shows how samples are classified.

Although various options for building abstract classification models are available, CTs represent a solution that combines high accuracy with a low complex tree structure [15] which enables the partitioning of even complex input space into ideally separated clusters. Such partitioning is obtained through recursive splits of the training data set (through the *Gini index* [15]) , hence a stopping criteria must be fixed such that the final tree structure includes only a subset of the available predictors, i.e., the most significant key features, hence, the ones that hold the most useful information to the solve the classification problem [145].

Quasi-exact logic functions through Classification Trees

In [147] the authors propose a novel ML-driven synthesis methodology that allows to describe generic Boolean functions through a representative subset of core expressions using Classification Trees (CTs). Obtained circuits are able to mimic Boolean functions to a certain degree of accuracy, hence the name quasi-exact logic functions.

According to the proposed methodology, a classification tree is trained using the truth table of a logic function output as a labeled training set. As reported in Figure 3.6 (top, from left to right), a single node of a CT can be seen as a Multiplexer (MUX) primitive. Therefore, any classification tree trained on Boolean samples can always be represented through a tree-of-MUXes. In addition, the proposed synthesis flow enables a smart hardware mapping on MUX-INV library by firstly translating the CT in a Binary Decision Diagram (BDD) which is efficiently reduced and ordered (3.6 - bottom, from left to right).

Experiments conducted on a subset of open-source benchmarks demonstrate that CTs are indeed able to cover rather complex Boolean functions with a very high degree of accuracy, 88% on average, still requiring 3× less area over conventional multi-level baseline circuits.

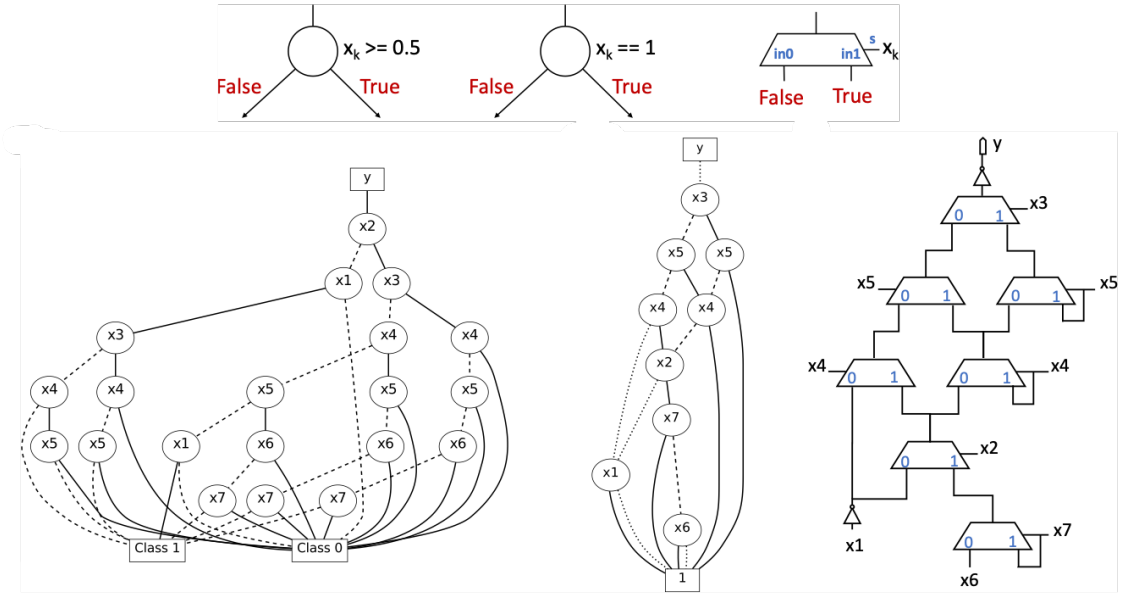


Figure 3.6: On the top, CT decision node representation and mapping: from left to right, original representation (i), node transformation (ii), and MUX mapping (iii). On the bottom, from CT to MUX-INV representation. From left to right, Original CT structure (i), reduced and ordered BDD representation (ii), MUX-INV implementation (iii). Solid and dashed lines represent true and false branches, respectively; dotted lines represent negated edges. [147].

3.3 Dynamic EAS: Run-time Tradeoff

3.3.1 Dynamic Voltage-Accuracy Scaling (DVAS)

As widely explained in the first part of this chapter, a dynamic energy-accuracy tradeoff brings an extra degree of freedom for power management. Among the state-of-the-art EAS techniques, Dynamic Voltage-Accuracy Scaling (DVAS) is one of the most representative and powerful strategy [102].

In its general embodiment, DVAS refers to dynamic techniques that allow voltage scaling through accuracy reduction without inducing timing errors. Compared to static EAS techniques which drop energy consumption by modifying the logic functions of the circuit building blocks, DVAS presents a run-time tunable energy-accuracy tradeoff which allows circuits to achieve larger energy savings when quality-of-results can be further sacrificed.

DVAS methodology can be easily explained by targeting arithmetic circuits, specifically an array multiplier (Figure 3.7). The energy efficiency of an array multiplier can be increased by truncating the input to reduce the circuits switching activity [52] [158]

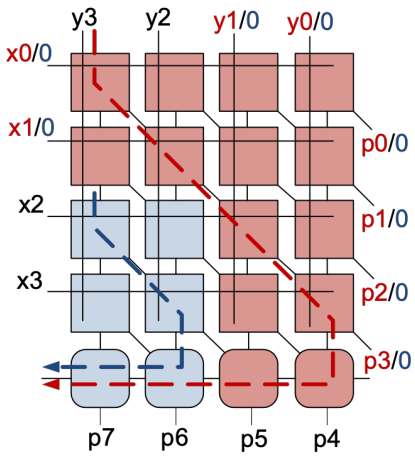
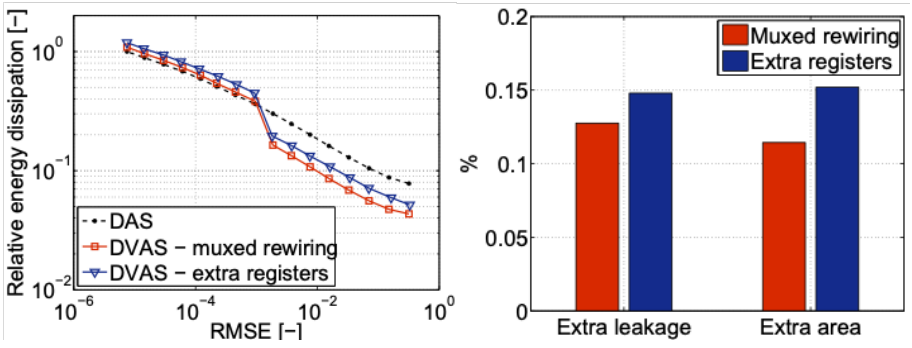


Figure 3.7: DVAS principle applied to an Array Multiplier [102].

at the cost an accuracy loss due to the quantization error introduced by the bit truncation, which by definition is bounded. However, using smaller bit widths not only reduces circuit activity but also shortens data paths enabling voltage scaling without timing violations. This principle, which represents the main intuition behind DVAS, is depicted in Figure 3.7: when the entire input bit width is used ($X = \{x_0, \dots, x_3\}$ and $Y = \{y_0, \dots, y_3\}$), the activity is high as the number of active gates is twenty, and the critical path is the longest possible (red dashed line). When only the 2 MSB's are used, only six gates are active and the critical path length drop (shorter blue dashed line), allowing an aggressive yet safe voltage scaling. For each input bit width a worst-case minimum voltage can be identified (at design-time), thus, the energy-accuracy tradeoff can be tuned by switching among different [*bit width*, *voltage*] operating points.



(a) Dynamic Energy vs. Accuracy (b) Static Energy and Area Overhead

Figure 3.8: Energy efficiency and Design overhead for a 16-bit DVAS multiplier [102].

DVAS can be generalized to pipelined systems by implementing EAS-oriented critical paths re-timing. This procedure is needed as in pipelined systems bit truncation

does not directly lead to shorter critical paths. Path re-timing is achieved by placing bypassable pipeline registers at strategic places of the data path. Two possible solutions to include by-passable registers in a digital design were implemented [102]: *(i)* extra by-passable registers placed in the specific path nets; *(ii)* muxed rewiring of specific signals to one single register, without placing extra registers. As reported in Figure 3.8, both techniques for pipelined DVAS introduces area/energy overhead which represents the major limitation of this dynamic EAS. Indeed, it has to be noticed that not all the design can undergo a re-timing/re-synthesis process with an "irreversible" modification of the circuit; also, this DVAS-oriented design reshaping includes dedicated optimization tools/algorithms that may be hard to integrate into commercial design kit.

3.3.2 Dynamic Voltage-Frequency-Accuracy Scaling (DVAFS)

The natural evolution of DVAS to further reduce digital ICs energy is represented by the *Dynamic Voltage-Accuracy-Frequency Scaling* (DVAFS) strategy [104]. It exploits the concept of *subword parallelism* [103] which increases the energy savings of DVAS by reusing inactive arithmetic cells at a reduced precision. Figure 3.9 reports a pictorial representation of this principle: a 4-bits array multiplier can be converted to a 4-bit subword parallel multiplier that can process two 2-bits subword operations per cycle. This allows, by keeping constant the computational throughput, to drop the operating frequency and hence scales voltage significantly below DVAS values. As a result, DVAFS is an EAS technique which simultaneously scales all run-time tunable parameters affecting energy consumption: *supply voltage* is dropped by shortening critical paths with bit truncation, *switching activity* is reduced with bit truncation, and at constant computational throughput *frequency* is cut via subword parallel operations.

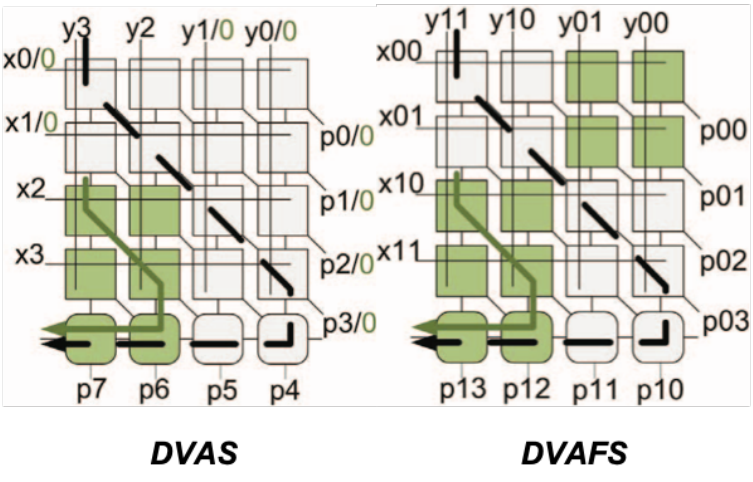


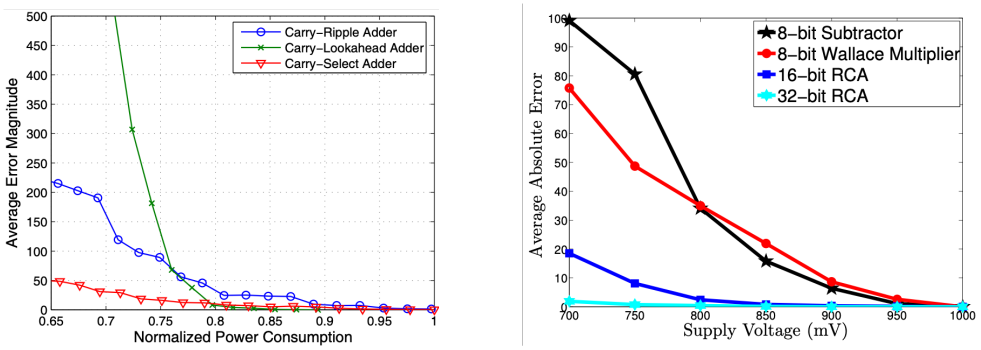
Figure 3.9: DVAFS principle applied to an Array Multiplier [104].

As for DVAS, the energy-accuracy tradeoff can be tuned by switching among different [$bit\ width, voltage, frequency$] operating points pre-defined at design time. To be noticed that DVAFS is based on a concept very similar to Dynamic Voltage-Frequency-Scaling (DVFS) [153]; however, in DVAFS, voltage and frequency are tuned on accuracy rather than throughput requirements.

Contrarily to DVAS, which can solely save energy in precision-scaled arithmetic blocks, DVAFS allows EAS in full system, including control units and memory, hereby shrinking energy overheads drastically at low precision. For this reason, DVAFS it has been applied to reduce energy consumption in highly parallelizable applications, like Deep Learning in embedded systems [103].

3.3.3 Dynamic Voltage Over-Scaling (DVOS)

The techniques mentioned above, DVAS and DVAFS, are conceived to keep the error magnitude under a specific bound (bit truncation worst-case error) avoiding timing fault occurrences and providing an acceptable average output accuracy. However, there are several works which try to exploit the intrinsic resiliency of different circuit architectures to aggressively scale the voltage allowing timing faults to occur and without imposing any maximum value to error. This EAS strategy is known in the literature as Dynamic Voltage Over-Scaling (DVOS) [65]. This technique extends the concept of dynamic voltage scaling beyond the critical voltage value imposed by the worst-case corner. If the circuit architecture is able to mask timing errors, scaling beyond critical voltage brings to significant energy savings without severe degradation of output quality. Obviously, logic circuits can run DVOS only after a design-time characterization of energy and quality-of-results.



(a) Adder comparison [91]. (b) Arithmetic functions comparison [100]

Figure 3.10: Arithmetic units under DVOS.

Researchers extensively investigated to identify the most suitable arithmetic unit for DVOS. [91] and [100] proved that Carry-Select-Adder provides the best energy–accuracy tradeoff under DVOS (Figure 3.10a); the reason behind this result lies in the

internal characteristics of this adder. Indeed, its architecture allows a graceful output degradation in case of timing errors. On the other hand, tree adders and subtractors fail as the supply voltage is scaled and, thus, the delay target is not met (Figure 3.10b, red and black plots respectively).

In [114], a characterization of various data representations under DVOS has been performed. It showed that radix-2 redundant binary can be considered the best EAS approach in systems with large data widths while 2's complement representation is more suitable in systems with small data widths.

By characterizing the most common computational kernels used in multimedia, recognition, and mining algorithms, [100] proposed design techniques to make the hardware implementations of these meta-functions behave more gracefully, i.e., to generate few or small errors, under DVOS. In Figure 3.11 is reported the results for *dot-product* meta-function implemented with both Ripple-Carry and Kogge-Stone adders optimized for DVOS.

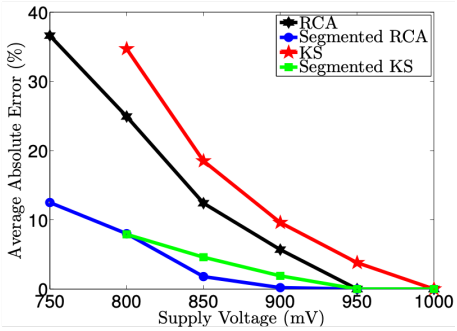


Figure 3.11: Error vs. Voltage plot for *dot-product* meta-function [100].

Also, the concept of Probabilistic CMOS, wherein each transistor and logic gate displays a probabilistic rather than deterministic behavior, was proposed as an energy efficient alternative to traditional deterministic computational models [113]. Such methods exploits of the statistical behavior of nano-scale devices and circuits and target probabilistic applications such as multimedia, recognition, and mining. Such applications perfectly fit for DVOS as they typically rely upon iterative and successive refinement techniques, which can mask errors introduced in the previous iterations. In other words, DVOS energy-accuracy characterization is performed on a probabilistic basis. Scalable Effort (SCE) [27] and Error Resilient System Architecture (ERSA) [84] are the most representative techniques of this approach.

As a final remark, DVOS energy-accuracy tradeoff can be considered sustainable as long as a probabilistic accuracy characterization proves a graceful output degradation when timing error occurs (as explained in [100]).

3.3.4 Dynamic Inferential Logic Circuits

Inferential logic circuits can also dynamically adjust the energy-accuracy tradeoff. The architecture of a Dynamic Inferential Logic Circuit (DILC), Figure 3.12, is a straightforward implementation of the Boolean formulation through Classification Tree (CT) described in Section 3.2.2. It consists of two main logical blocks: *(i)* the Inferential Unit (IU), which implements the CT function I; *(ii)* the Supervisor Unit (SU), which restores the output of F when I yields to an incorrect prediction. As shown in [146], DILC can work in two operating modes regulated at run-time.

Quasi-exact mode: the IU evaluates the input X; its output is similar to the Boolean function F. In this run-mode, accuracy can be sacrificed for energy savings.

Exact mode: both the IU and the SU evaluate the input X. Two are the possible outcomes: *(i)* the input pattern belongs to the set of misclassified patterns, hence the output inferred by the IU is wrong, and the SU redirect the 1's complement of IU toward the primary output; *(ii)* the input pattern belongs to the set of correctly classified samples, hence the value inferred by IU is propagated to the output. This operating run-mode can be dynamically enabled when accuracy scaling is not an option.

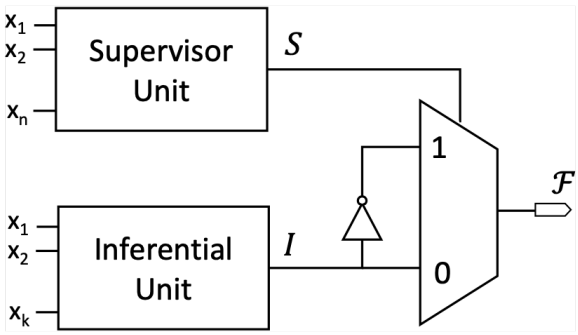


Figure 3.12: Dynamic Inferential Logic Circuit architecture [146].

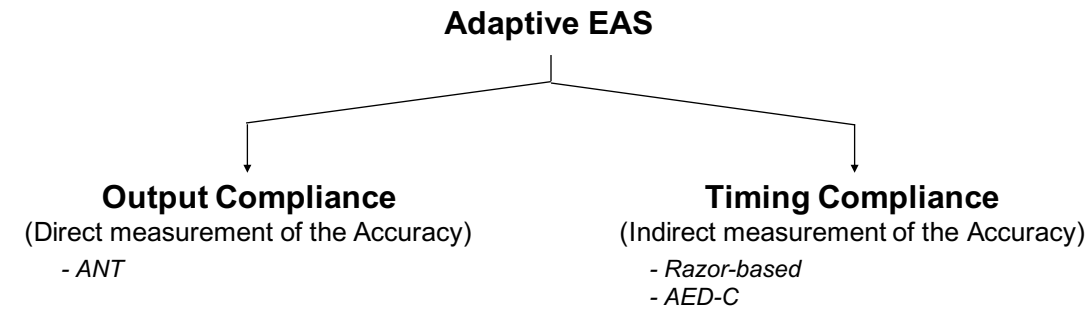


Figure 3.13: Adaptive EAS classification depending on the feedback loop nature.

3.4 Adaptive EAS: Beyond Dynamic Tradeoff Limitations

Adaptive EAS is the speculative evolution of the dynamic EAS approach as instead of switching among conservative operating points pre-defined at design-time (please refer to DVAS/DVAFS), the adaptive approach let the input workload, measured through on-chip monitors at run-time, to drive the circuit to the optimal energy-accuracy tradeoff.

Following a more formal definition, the techniques in this class implement a closed-loop EAS, i.e., an approach which adaptively fine-tunes the energy-accuracy tradeoff by directly measuring quality metrics on-chip (i.e., computation-compliance information), at run-time. For the sake of clarity, the nature of the feedback information is strictly related to the input workload, i.e., the flow of input data, and to the topology of the circuits, i.e., to the logic paths distribution. According to the works proposed in the literature, the feedback quality metrics can be a direct measurement of the output accuracy or an indirect estimation of the accuracy value. Specifically, as shown in Figure 3.13, Adaptive EAS can be driven by *(i)* the *Output Compliance*, i.e., the computation of the quality-of-results (e.g., the error magnitude), or *(ii)* the *Timing Compliance*, i.e., an evaluation of the output accuracy using circuit timing information (e.g., the timing error rate). Let us analyze this classification by reporting the most representative adaptive AES techniques:

- *Adaptive EAS driven by Output Compliance*
  This approach is usually implemented through on-line monitors that perform a direct measure of the output error magnitude. The most representative strategy for this class of techniques is Algorithm Noise Tolerance (ANT) [57, 56]. The main idea of ANT is to shift the correction of the timing errors to a lightweight replica of the circuit, the estimator. The output produced by the estimator is compared with the output of the main circuit by a decision block that computes their difference. When such difference exceeds a given threshold (pre-determined at design time), the error-control block forwards the estimator output to the main output of the circuit, such that the output error can be mitigated. The most relevant benefit

of ANT strategies is that an upper bound on the error magnitude can be imposed. However, as shown in the next session, this comes at the cost of considerable design overhead.

- *Adaptive EAS driven by Timing Compliance*
  This strategy adapts the system voltage w.r.t. the delay of the circuit paths sensitized by the input workload, i.e., to the *timing error rate*. As shown in the literature, the timing error rate is usually measured through Error Detection and Correction strategies which used in-situ timing sensors, such as Razor, to controls the voltage scaling [76] [126]. There might be specific sequences of input patterns that push the supply voltage so down that some of the longest paths could even bypass the detection mechanism leading to potential error propagation in the fanout logic; such cases, called *error miss-detections*, represent the primary source of accuracy loss. In [126], all detected errors are corrected, but the occurrence of error miss-detection contribute to quality-of-results degradation. On the contrary, in [76] errors are only counted, but no error recovery is performed, directly affecting the output quality. In general, if the error rate is lower than a user-defined threshold, Vdd is scaled; otherwise, the supply voltage is raised back to ensure safe operation. In this way, the circuit tends to work close to the edge between error-free and erroneous computation.
  A novel strategy, which is the core of the next Chapter, leverages on Approximate Error Detection-Correction (AED-C) [127]. This technique implements EAS using the error detection coverage as a knob: a low error coverage accelerates supply voltage scaling thus to achieve larger energy savings at the cost of quality of result; a high error coverage lessens the voltage scaling leading to better output quality at the cost of lower energy savings. The AED-C mechanism is built upon *elastic timing monitors*, Razor FFs augmented with a tunable detection window and hardened with the aid of a dynamic short-path padding technique.
  Opposite to ANT, in timing-compliance EAS approach, the maximum error value is not deterministic, as it can be arbitrarily high. The accuracy loss is kept bounded through a speculative mechanism based on probabilistic rules that guarantee the longest paths,i.e., the ones more prone to run into miss-detection, are rarely activated keeping the long-term average accuracy loss limited. This aspect of adaptive EAS is detailed at the end of this chapter.

The following sections discuss more in detail ANT and Razor-based adaptive solution, analyzing their advantages/drawbacks and field of applications; but first, the next section deals with *Adaptive Voltage Over-Scaling* (AVOS) an aggressive voltage scaling strategy for error-resilient applications widely used in these techniques.

3.4.1 Adaptive Voltage Over-scaling (AVOS)

As extensively explained in Section 3.3, Dynamic AES approaches complement voltage scaling with a concurrent adjustment of the accuracy. For example, in the case of DVAS, voltage scaling is compensated by decreasing accuracy; in DVAFS case accuracy-frequency values are scaled together. As a result, these EAS systems always run in a "stable" voltage-accuracy, or voltage-accuracy-frequency, operating point, where the critical-path delay always match the clock period. This conservative design imposes that the circuits to always meet the timing constraints by taking into account safety margins due to worst-case temperature/process variations. Even in DVOS design techniques, where timing errors occurrence are allowed during voltage-accuracy scaling, graceful timing constraint violations (few or small errors)must be ensured by a design-time characterization, hence maintaining a significant degree of conservatism.

Such conservative EAS approach implies that the majority of circuits cannot exploit the full potential of voltage scaling at the post-fabrication time, preventing significant energy savings. In other words, even though Vdd could be further scaled without compromising the system's compliance, it remains in worst-case operating condition since the power management policy has no run-time feedback information on the circuit health.

Adaptive Voltage Over-Scaling (AVOS) is an adaptive power management system for error-resilient applications [76]. In AVOS the circuit operating voltage is scaled adaptively depending on the circuit run-time behavior leaving frequency unchanged, i.e., without system's performance reduction. Voltage is selected dynamically among a set of possible values by a power management unit driven by sensed on-chip compliance information. Conservative assumptions adopted in Dynamic EAS techniques (i.e., worst-case process/temperature variations) are not necessary anymore; this enables an aggressive voltage scaling with larger energy savings.

3.4.2 Algorithm Noise Tolerance (ANT)

Implementation

The basic principle underlying the ANT technique is to accept errors as long as the output degradation due to Vdd scaling remains below a given noise threshold [57, 56]. As depicted in Figure 3.14, a typical ANT architecture consists of the main circuit coupled with its own lightweight replica, known in the literature as Reduced Precision Replica (RPR) [137][136].

Such replica is approximated through arithmetic precision scaling, namely dropping some of the LSBs and it serves as a ground reference for the assessment of the output quality of the circuit during the voltage scaling. When AVOS power management is applied to this technique, timing faults do appear in the main circuit at first, whereas the intrinsically faster replica runs fault-free for lower Vdd. The supply voltage is regulated by monitoring the arithmetic error, which is given as the difference between the output

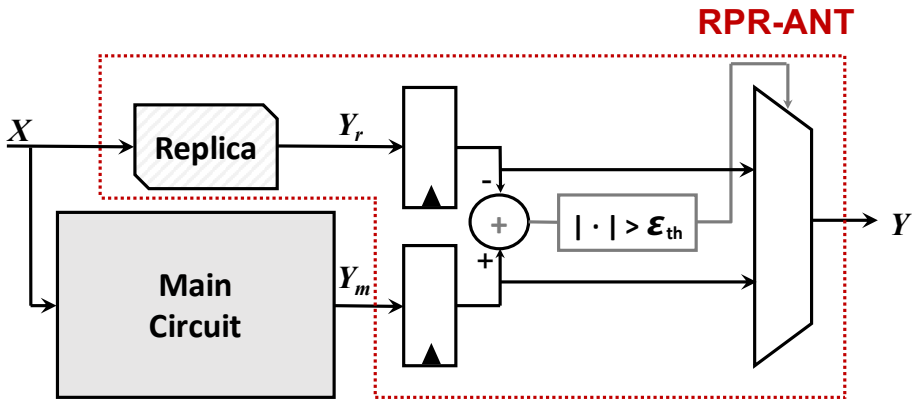


Figure 3.14: RPR-ANT Block Diagram.

of the main circuit and that of the replica. The detection unit flags an event when the difference overcomes a pre-defined threshold ($\mathscr{E}_{th}$). In such case, the replica's output is forwarded towards the main output of the circuit. An important aspect is that the output error is bounded by the arithmetic precision of the replica.

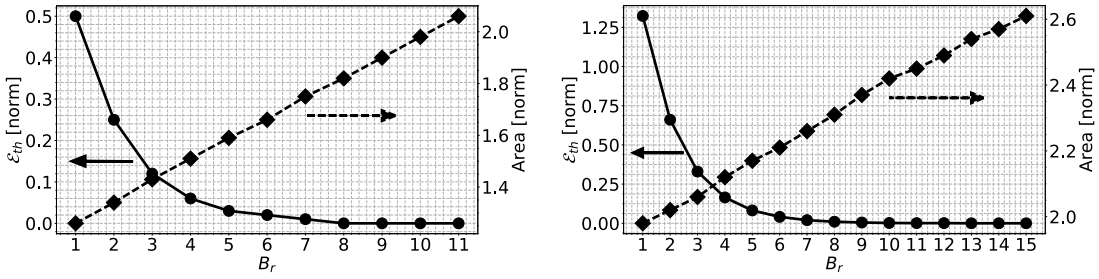
Design and Area overhead

The design of the replica circuit and that of the control circuitry introduces some overhead which should be carefully weighted against the actual savings brought by AVOS. The main challenge is to limit the area and delay penalty while guaranteeing the desired output quality. As a preliminary analysis, the design characterization for two benchmarks is reported : FIR and IIR digital filters processing a sequence of three different baseband audio signals. We implemented an entire set of RPR-ANT circuits by changing the precision of the replica circuit, namely reducing the input bit-width from 1 to $B-1$, where B is the bit-width of the original circuit. For each implementation, the error threshold ($\mathscr{E}_{th}$) of the decider unit over the experimental testbench input patterns is computed by applying the formula explained in [137]:

$$\mathscr{E}_{th} = \max_{input\ patterns} \mid y_o[n] - y_r[n] \mid \tag{3.1}$$

with y_o as the error-free output, and y_r the output of the replica circuit. Fixing $\mathscr{E}_{th}$ in such a way ensures that the output of the circuit Y is equal to the main circuit output Y_m in the absence of timing errors.

Figure 3.15 shows the trend of $\mathscr{E}_{th}$ normalized over the output range of y_o ($y_{range} = | max(y_o) - min(y_o) |$) and the Area overhead of the architecture versus the number of the Replica bits (B_r). As expected, the decision threshold increases exponentially when B_r drop, as shown in [137]. The area overhead w.r.t. the baseline circuit increases almost linearly. For the FIR filter, the area ranges from 1.26× to 2.06×, respectively for B_r equal to 1 and 11. In the IIR filter case, even with B_r = 1 the area is 1.98× larger than the



(a) FIR filter (b) IIR filter

Figure 3.15: Error Threshold and Area overhead vs. Replica circuit bit-width.

baseline filter, and peaks up to 2.61× for B_r = 15. Such large area overhead is due to the internal characteristics of the filters, i.e., the feedback branches of IIR in direct form I. These results are in line with the conclusions achieved in[112], where for a set of benchmarks {RM-CORDIC, FIR Filter, Square Root Unit} the area overhead reaches the peak of {82%, 128%, 143%} w.r.t. the baseline circuit.

The RPR-ANT paradigm implies the availability of a quantitative measure of error, which is straightforward for (stand-alone) arithmetic circuits, while it becomes difficult for other kinds of circuits (embedded into more complex systems perhaps). Finally, the area overhead due to the replica circuit is not negligible; despite the many works on the field, the design of a faster and smaller circuit replica still remains an open issue, especially for non-arithmetic circuits. Hereafter, for the sake of readability, the RPR-ANT strategy is simply labeled as ANT.

3.4.3 Error Detection and Correction schemes for Adaptive EAS

As explained in Section 2.5, two main design strategies can be used for always-correct applications. Top region of Figure 3.16 reports them. *Worst-case margined* design (top-left region), which add sufficient clock-time margin to compensate delay variations. Conventional *Error Detection and Correction* (EDC) methods (top-right region) which sense, at run-time, the timing margin by detecting and recovering all local timing errors (e.g., in-situ), such that the system can be tuned to operate at the margin edge, i.e., at nearly-zero timing slack. In the class of those strategies that are employed for error-resilient applications but do not integrate any error detection mechanism (Figure 3.16, bottom-left region), both static and dynamic EAS strategies can be reported by way of example.

On the other hand, Adaptive EAS strategies based on EDC (bottom-right region) is a speculative extension and generalization of conventional Energy scaling EDC methods to error-resilient applications [1]. As EAS does not have to ensure full error coverage, the traditionally large area/energy overhead of error detection and recovery in conventional EDC (refer to section 2.5.1) may be reduced by using a simpler error management

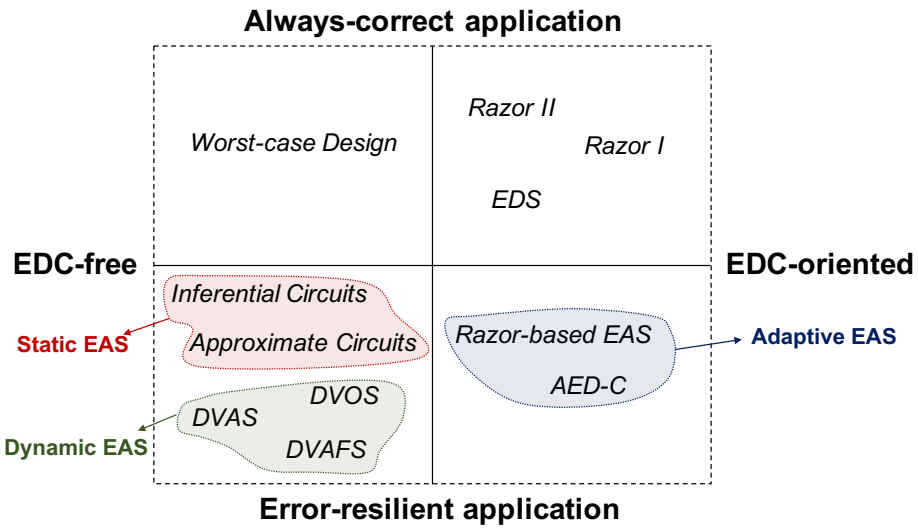


Figure 3.16: EDC for always-correct vs. error-resilient applications.

circuitry. Thus, these techniques are able to sense the timing errors but they may run into miss-detected errors to deliver a more aggressive voltage scaling, thus, larger energy savings. The timing error rate is the statistical tool that drive the circuit to the optimal energy-accuracy tradeoff, while the long-term average accuracy loss is kept within a reasonable range by the probabilistic rule for which longest paths, the ones that cause miss-detections activate rarely.

In the following sections a brief introduction to EAS methods based on Razor is reviewed.

Implementation of Razor-based EAS

According to the literature EDC-based Adaptive EAS is mostly implemented through Razor-style in-situ timing sensors which enable aggressive power management as Adaptive Voltage Over-Scaling. A lightweight Razor FF error detection circuitry has been used in [76], to implement an energy-accuracy scalable FTC1/DXT1 decoder. Razor FFs are labeled with error significance weights and bit-wise ORed to yield the overall error degree which is compared to a pre-defined threshold to drive the voltage scaling. Most importantly, no error recovery is performed. Also, no short-path races fixing measures have been implemented. The detection window DW, i.e., the clock delay to shadow FF, has been chosen to match the minimum delay of the shortest path at the input of Razor FF. Then, DW is changed dynamically with the supply voltage to detect as much timing violations as possible.

Experiments on the FTC1/DXT1 decoder show that AVOS achieves significant energy reduction while the additional error is negligible compared to the quality loss due to lossy compression (Figure 3.17). The major issue related to this work is that the error detection mechanism cannot work under process variations; this makes this approach

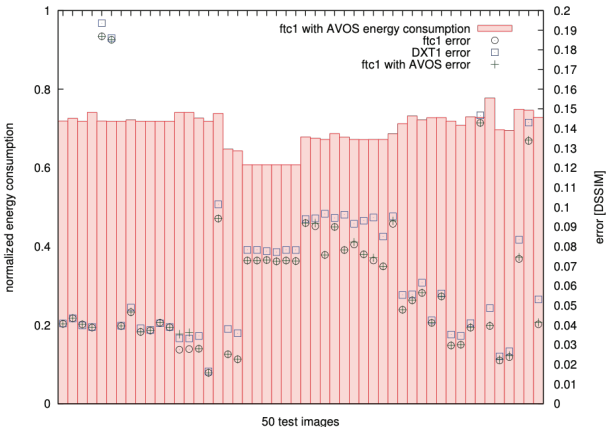


Figure 3.17: Energy vs. Error over 50 test images in [76] experimental setup.

hardly to be implemented at circuit-level. Indeed, the analysis reported in [76] are not the results of a gate-level netlist functional simulation, but only a probabilistic estimation. For these reason this technique is not employed for a comparative analysis with the core of this dissertation, i.e., the Approximate Error Detection-Correction (AED-C).

Authors in [126], proposed *Early Bird Sampling* (EBS), a Razor variant that enables AVOS for error-resilient low-power circuits. The EBS allows to solve the problem of short-path races through a Tunable Delay Line at the Razor FF input bypassing tedious hold-time fixing design stages. Also, EBS reduces design overhead exploiting a local logic-masking mechanism for error correction. The simulation on a set realistic benchmarks shows that EBS can implement a more efficient EAS mechanism compared to the standard Razor hold-fixing through buffers. EBS is discussed more in detail in the next chapter.

3.4.4 Adaptive EAS and errors characterization

As explained in the sections above, ANT architecture allows a direct control on the error magnitude and is able to keep the output degradation under a specific threshold fixed by the approximation mechanism. On the contrary, the timing-compliance speculation approach cannot impose any direct boundary to the magnitude of errors, as the latter are the results of miss-detected timing violations. However, two mechanisms based on probabilistic rules limit the average accuracy drop: *(i)* the voltage scaling is kept to safe values, i.e., that ensure no dramatic accuracy degradation, by the timing errors rate due to the most active path; *(ii)* the longest paths, i.e., the ones more prone to run into miss-detection (thus leading to quality degradation), are rarely activated guaranteeing limited long-term average accuracy loss.

Following a more formal definition given in [26, 109], the errors introduced by the voltage scaling in ANT remain "small" in magnitude (as it is bounded by the precision of the replica circuit) but quite frequent. Thus, ANT can be associated to the class of *Fail*

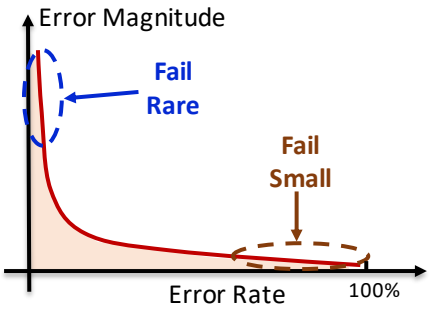


Figure 3.18: Adaptive EAS and error characterization.

small applications (bottom-right region in Figure 3.18). On the other hand errors, the EAS based on timing compliance belongs to the class of *Fail Rare*. The error magnitude due to miss-detection is arbitrarily high (the long timing paths that fall in violations are commonly on the MSB of the output, especially in arithmetic circuits), but very infrequent (long timing paths are activated rarely) guaranteeing an acceptable average accuracy (top-left in Figure 3.18). This classification is extensively validated in the next chapter.

Chapter 4

Adaptive EAS via Approximate Error Detection-Correction

As extensively discussed in Section 3.4.3, Adaptive Energy-Accuracy Scaling (EAS) can be managed by an estimation of the output accuracy obtained by checking the circuit timing-compliance through conventional Error Detection and Correction (EDC) schemes. As the timing errors rate drives the voltage scaling, reducing the error detection coverage of standard EDC monitors enables an aggressive voltage reduction, thus, larger energy savings, as long as the average accuracy degradation is kept bounded by the probabilistic assumption for which the longest paths, hence, miss-detected timing errors, are rarely activated. Also, as EAS does not have to ensure full error coverage (i.e., admit error miss-detections), the traditionally large area/energy overhead of error detection and recovery in standard EDC may be reduced by using a simpler error management circuitry.

This chapter introduces *Approximate Error Detection-Correction* (AED-C), an error management scheme suited to adaptive EAS for error-resilient applications. The AED-C mechanism tackles standard EDC complexity in *(i)* the short-paths race management and *(ii)* detection mechanism to introduce a more flexible and efficient energy-accuracy scalability. As a result, AED-C is implemented by in-situ *elastic timing monitors*, i.e., *Razor* flip-flops augmented with two innovative features:

1. *Dynamic short-path padding*, obtained through a technique referred to as *Early Bird Sampling*, which ensures low area overhead and efficient voltage scaling by overcoming the limitations of the Razor static short-path padding based on buffers insertion procedure;

2. *Tunable Error Detection* capability, i.e., timing sensors with a tunable Detection Window mechanism to control the energy-quality scaling tradeoff.

Inspired by the working principle of Approximate Computing, AED-C enables EAS using the error detection coverage as a knob: a low error coverage accelerates supply

voltage scaling thus to achieve larger energy savings at the cost of Quality-of-Result (QoR); a high error coverage lessens the voltage scaling leading to high QoR at the cost of weaker energy savings.

This chapter firstly discuss the features of the proposed short-pah padding strategy, the *Early Bird Sampling*, reporting its figures-of-merit compared to state-of-art Razor. Specifically, it shows how the problem of short-path races can be solved bypassing severe hold-time fixing stages based on buffer insertions and presents a low-overhead local logic-masking mechanism for error correction. Then, a section is dedicated to the *Tunable Error Detection* (TunED) technique which explains the implementation of timing sensors embedding Tunable Detection Window mechanism.Finally, AED-C strategy is disclosed at architectural-/circuit-level along with the EDA tools used to implement it. AED-C driving a dual-mode Adaptive Voltage Over-Scaling (AVOS) is simulated over a representative set of circuits for image/audio processing (e.g., DCT, FIR/IIR digital filters) providing a pros&cons analysis and a comparison with the state-of-art Razor. The collected results show that AED-C substantially reduces the average energy-per-operation (up to 44.7% savings w.r.t. Razor-driven AVOS) and the area overhead (3.3% vs. 62.0%), still guaranteeing reasonable accuracy. As an example, when applied to a real-life application, i.e., a DCT integrated into a JPEG compressor, AED-C shows 51.9% energy savings (w.r.t. a baseline DCT implementation) ensuring a PSNR of 48.45 dB (w.r.t. baseline JPEG images). The chapter is closed by a comparative analysis between AED-C and another technique that belongs to the Adaptive EAS class, i.e., Algorithm Noise Tolerance (ANT), giving final considerations on AED-C benefits and limitations.

4.1 Early Bird Sampling: a Short-Path Free Error Detection Strategy

Razor is a milestone in the field of Error Detection and Correction (EDC) strategies for low-power operation. Despite the impressive level of maturity, its application on circuits other than pipelined processors still remains an open issue. Firstly, the error detection mechanism relies on special flip-flops (FFs), the Razor-FFs, whose use imposes heavy hold-time fixing and large circuit area/power overheads; secondly, the error correction is performed through instruction replay, a practice that is not available (or very expensive to implement) in generic circuits.

Early Bird Sampling (EBS) is a Razor variant that applies to low-power sequential circuits. The EBS allows to *(i)* solve the problem of short-path races bypassing tedious hold-time fixing design stages, *(ii)* reduce design overhead exploiting a local logic-masking mechanism for error correction. As a key feature, EBS enables the Adaptive Voltage Over-Scaling (AVOS), particularly suited for ultra-low power error-resilient applications.

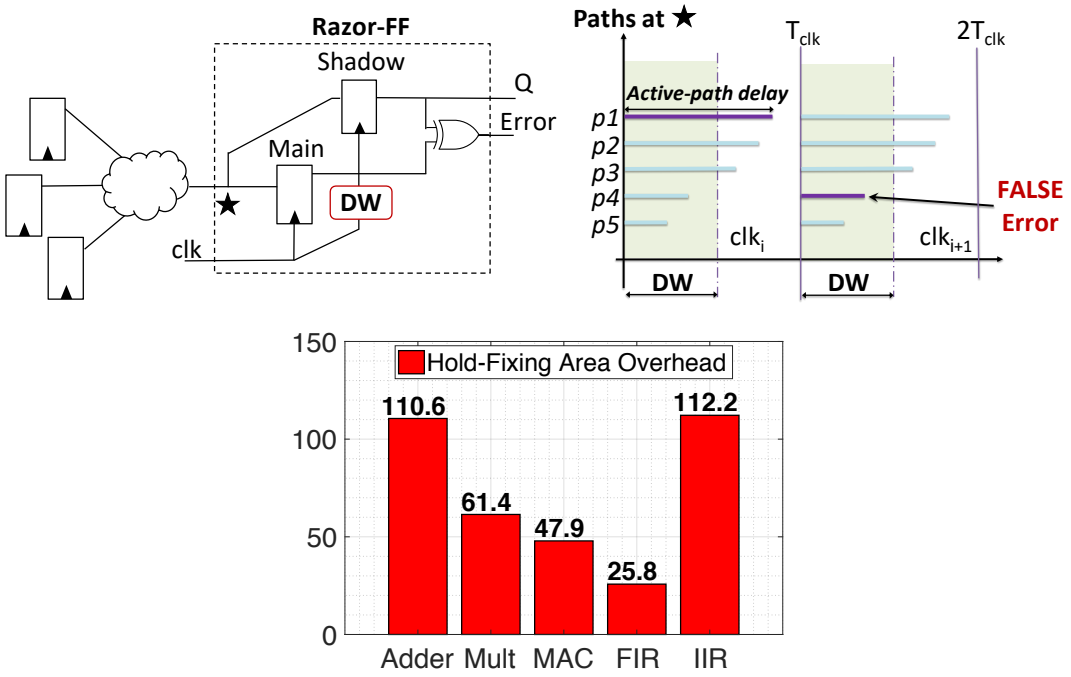


Figure 4.1: Razor-FF implementation (top-left); Short path race (top-right); area overhead due to short-path padding (bottom), % w.r.t. baseline circuit [126].

4.1.1 On the Limitations of Razor Scheme

In its more general embodiment, an adaptive IC executes an EDC scheme where the occurrence of timing errors (typically due to set-up time violations) is flagged to the control management unit that eventually operates a recovery mechanism. If the circuit is healthy, i.e., no errors, power consumption is optimized by means of some low-power knob, e.g., voltage and frequency scaling, body-biasing, or a mix of them.

Among the many EDC solutions appeared in the recent literature, *Razor* [38][31] still represents the state-if-the-art. The error detection is implemented by replacing standard flip-flops (FFs) with special FFs, a.k.a. *Razor-FFs* (Figure 4.1 top-left), that sample logic signals at two different instants of time: first, at the rise edge of the clock, then, after a predefined timing window, the so-called *Detection-Window* (DW). The two time-skewed samples are stored by two different FFs, i.e., the *main* flip-flop and the *shadow* flip-flop, and then compared through a XOR gate for parity check. A parity match implies the absence of errors and the availability of some timing slack, whereas a mismatch implies a faulty computation that is then recovered through some correction mechanism. To notice that Razor has been conceived for pipelined processors, hence, error recovery is accomplished through instruction replay.

Although Razor is considered a milestone in the scientific community, it shows intrinsic limitations that prevent its use on sequential circuits other than pipelined processors. The main reasons are two (described below in criticality order).

1. Short-path races.
While processors show a relatively small number of end-point FFs (the stage registers of the pipeline) most of which having a regular timing path distributions, generic sequential circuits have many FFs usually driven by logic cones with timing path distributions that seriously complicate the timing closure during logic synthesis. To better understand this critical aspect, one should consider Razor-FFs suffer the so-called *short-path race*. As per their internal structure (Figure 4.1 top-left), Razor-FFs cannot make distinction between the activation of a short-path within the DW and the activation of a long-path beyond the clock edge. This may cause "false" error detections. As depicted in Figure 4.1 (top-right), the value sampled in the main FF at (T_{clk}) may differ from that sampled in the shadow FF at ($T_{clk} + DW$) due to a short-path activation (p4); the error flag is then raised even if there is no timing violation.

In order to avoid overlaps between short- and long-paths, a common design practice is to apply a static short-path padding [38] [31]. It is a constrained hold-time fixing procedure (*short-path padding* hereafter) where buffers are selectively inserted in the logic cones such that the minimum arrival time of any logic path is shifted beyond DW (usually 50% of the clock-period); short paths delaying is done while keeping the longer timing paths untouched. The side effects are many. Firstly, long buffer chains induce huge area penalties. As a preliminary result, Figure 4.1 (bottom) shows the area overhead due to short-path padding for the set of circuits we used as benchmarks: the worst case is 112%. Secondly, when the timing constraint on the long-paths is tight, short-path padding tries to reach timing closure by means of Boolean transformations and heavy circuit topology modifications that (*i*) further increase area (*ii*) reshape the path distribution with a negative impact on Vdd scaling efficiency (more details in the experimental section). Finally, the insertion of long buffer chains exacerbates the timing unpredictability due to PVT variations when the circuit works at ultra-low voltage, e.g., near-threshold [70]. In this case, for a simple test-case made up of three 16-bits multipliers, the area grows by 2.1x, while energy per operation increases up to 81% w.r.t. the baseline circuit.

Even assuming the overhead of short-path passing could be brought below a reasonable threshold, it still remains static, thereby preventing the implementation of any tunable strategy as the target of this chapter, the *Approximate Error Detection-Correction*, requires.

Some previous works, e.g., [31, 80], addressed some of the above issues by means of a reduced DW obtained through a duty-cycled clock (duty-cycle $< 50\%$). These solutions have proven to be effective only in some specific case of study. The authors of [161] substitute buffers with a fine-grained allocation of dummy loads (spare cells and dummy metal). The integration of this methodology in commercial tools might result too complex, or not compliant with design rules checking (DRC).

A solution to avoid the insertion of hold-fixing buffers while keeping a 50% duty-cycled clock is proposed in [29]. The authors introduce a short-path padding methodology based on Latch-Insertion; for each pipeline stage, logic cones are split into two parts using Negative-phase transparent latches; during the high phase of the clock, latches prevent short-path transitions at the inputs of the Razors-FFs. A potential limitation is that the arrival time of the latched block can be altered by process variations leading to an increase of the error rate. Also, the area overhead of latches may run out of control when dealing with random circuits.

Finally, Bubble Razor [42] was proposed to break the dependency of short-paths from DW and hence to achieve a lower short-path padding overhead. This technique requires flip-flops to be converted into two-phase timing latches, a solution that is not fully supported by commercial design flows. Moreover, the control flow for generating/propagating the bubble might be complex, with large overhead that limits the application of the technique.

For AED-C strategy implementation, none of the above solutions can address the requirement of a *dynamic* short-path padding strategy.

2. Correction through functional redundancy.
While pipelined processors offer an easy path to error correction, i.e., instruction reply, implementing the same mechanism on sequential circuits would require a too complex FSM rewind. Hence, alternative circuit strategies are needed [154]. Unfortunately, the design overhead of such correction circuitry might substantially affect the gain brought by adaptive power management.

These two class of issues make Razor implementation very hard, often impractical, to be adopted in low-power ICs. Also, most of the attempts made to generalize the Razor technique have turned out to be too costly. Next sections deal with these issues proposing a lightweight EDC strategy, called *Early Bird Sampling* (EBS), that addresses the short-path races issue through the insertion of Tunable Delay Lines (TDLs) shared among *all* the paths flowing onto the same end-point. The result is that of depleting the DW from short-paths and avoid false error detections without incurring any significant design overhead. To notice that the availability of tunable delays enables post-silicon variability compensations.

Also, as shown below, EBS enables an efficient AVOS scheme as voltage scaling is able to follow the actual activation of the longest path (rather than the worst-critical path identified at design time). This leaves room to a more aggressive power management well suited for error-resilient applications.

4.1.2 Dynamic Short-Path Padding through Early Bird Sampling

The Early Bird Sampling (EBS) technique has been conceived with a twofold objective in mind: (*i*) reduce traditional design overhead imposed by Razor system, while (*ii*)

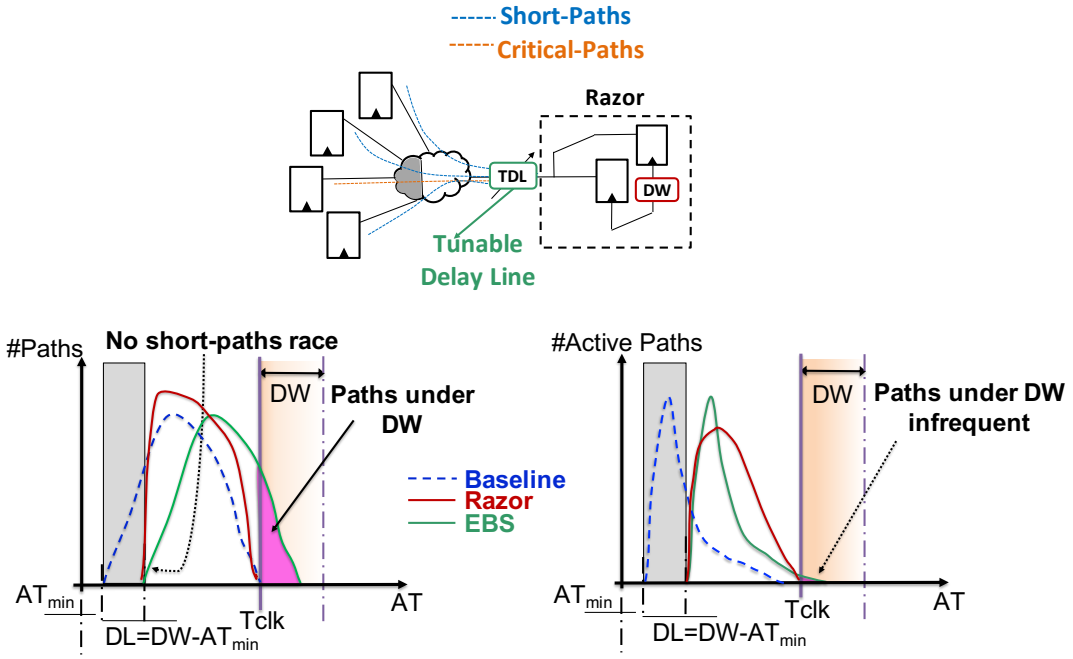


Figure 4.2: Early Bird Sampling at a timing critical end-point: circuit implementation (top); static (bottom-left) and dynamic (bottom-right) timing paths analysis. Plots are illustrative and do not refer to a specific case, rather, they show typical distributions observed on generic circuits.

maintaining those intrinsic characteristics of the circuit that enable an efficient implementation of AVOS.

A schematic representation of the EBS circuit is given in Figure 4.2 (top). Tunable Delay Lines (TDLs) are inserted just before the critical end-points of the circuit. Those end-point are equipped with a variant of the Razor-FF (more implementation details provided later in the text). The propagation delay of the TDLs can be tuned at run-time such that the arrival time of the shortest path (AT_{min}) is delayed beyond the detection window (DW) of the Razor-FFs. Indeed, the delay of a TDL is given by:

$$TDL = DW - AT_{min}; \tag{4.1}$$

This prevents the activation of short-paths within the detection window, and so, races with long-path in setup time violation, i.e., "false" error detection. For the sake of clarity, we assumed that a TDL is tuned during post-fabrication stage, when also the nominal T_{clk} can be properly set such that no paths can be delayed beyond the DW in nominal operating conditions. Each critical end-point comes with its dedicated TDL. To be also noticed that the tunable delays enable post-silicon compensation on the short-paths (out of the scope of this work).

The EBS strategy can be seen as a "weak" short-path padding optimization procedure where the set-up constraints are not taken into account. Indeed, a TDL does not

delay short-paths only, actually, it evenly affects all the paths in its fan-in cone. The longest paths may thereby suffer early sampling, hence the name *Early Bird Sampling*. The relaxation of the timing constraints is the key for a lightweight implementation of the error-detection mechanism.

To better understand the working principle of EBS, Figure 4.2 (bottom-left), provides a comparison among the static path distributions at a critical end-point for three different circuits implementations: (*i*) a generic circuit after synthesis, the *Baseline* (blue dashed line), (*ii*) the circuit after standard short-path padding optimization, *Razor* (plain red line) (*iii*) *EBS* (plain green line). Short-path padding reshapes the path distributions guaranteeing that all paths are beyond the detection window, namely, outside the gray area in Figure, while maintaining the longest path delay unchanged. By contrast, the effect of EBS is to shift the whole timing distribution, hence, some paths move beyond T_{clk} (purple area).

At first glance, this issue may be seen as a potential impediment. However, a more accurate analysis reveals that the problem is less relevant from a practical viewpoint. EBS exploits the fact that for real-life workloads the activation probability of long-paths is usually pretty low. This feature, shown by the majority of digital circuits, suggests that latent faults on long-paths are rarely excited. Experimental results give evidence of such empirical rule of thumb, which can be inferred by probing the arrival time of timing end-points during workload execution, i.e., through a dynamic timing analysis. Figure 4.2 (bottom-right) plots the dynamic path distribution for a typical workload run on three different implementations of the circuits: baseline, Razor, EBS. As a matter of fact, the number of violating paths is much lower than those estimated using a worst-case static timing analysis (purple area in Figure 4.2 bottom-left).

The most interesting aspect is that EBS does not alter the shape of the distribution (both static and dynamic); referring to the plots in Figure 4.2, the green line is a copy of the dashed line, just shifted on the right. This allows to preserve the intrinsic characteristics of the original circuit, thus enabling a more efficient voltage scaling. The same is not for Razor, where a *path compression* resulting from short-path padding optimization substantially increases the number of "quasi-critical" paths (i.e., more active paths skewed towards T_{clk}, red curve above the green one) as shown in Figure 4.2 (bottom-right). As a side effect, even small voltage variations would bring a large number of paths beyond T_{clk}, therefore triggering more timing errors. As a result, power-management techniques, and AVOS in particular, might have fewer margins to operate; this impacts the voltage scaling efficiency and the potential energy savings.

The intuition behind EBS is that a simple delay shift is less invasive and more suitable for aggressive voltage scaling. The truthfulness of this principle is proven by experimental tests, which reports the results achieved by an aggressive Adaptive power management like AVOS.

4.1.3 Implementation Details

Tunable Delay Line (TDL)

Different implementations of TDLs have been proposed in literature; as the modeling of a TDL is out of the scope of this dissertation, we opted for the solution presented in [165]. It consists of a pair of inverters with a voltage-controlled variable load between them; the load is a transmission gate whose ON-resistance is controlled by V_{delay}, as shown in Figure 4.3. Such solution allows to cover a wide range of delays with a limited area overhead. The main drawback is that an extra power grid is needed for the distribution of V_{delay}. An alternative solution is to use tunable buffers adopted for on-line clock-skew compensation [23].

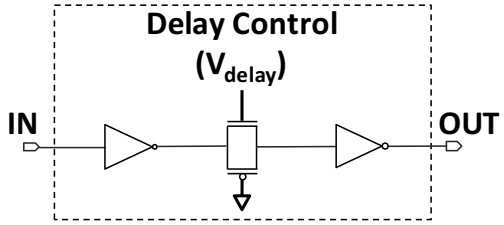


Figure 4.3: Tunable Delay Line (TDL) implementation [165].

Error Detection-Correction

The EBS detection and correction mechanism is implemented using standard Razor-FFs [39] augmented with a logic masking circuitry [154], Figure 4.4. Hereafter, we refer to this architecture as *Razor-Logic-Masking* (Razor-LM). A polarity change at the input of the main flip-flop after the rising edge of the clock implies some long-path is violating the timing constraint, i.e., a timing error. This event is flagged through the XOR gate that runs a parity check between the signals at pins D_{FF} and Q_{FF}. The error flag is sampled in a shadow latch triggered on the fall edge of the clock.

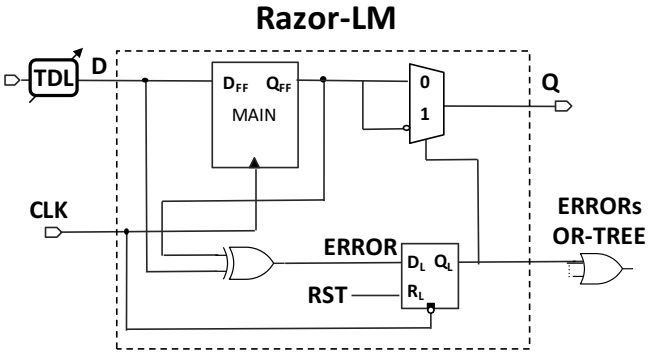


Figure 4.4: Error detection and logic masking circuitry in EBS.

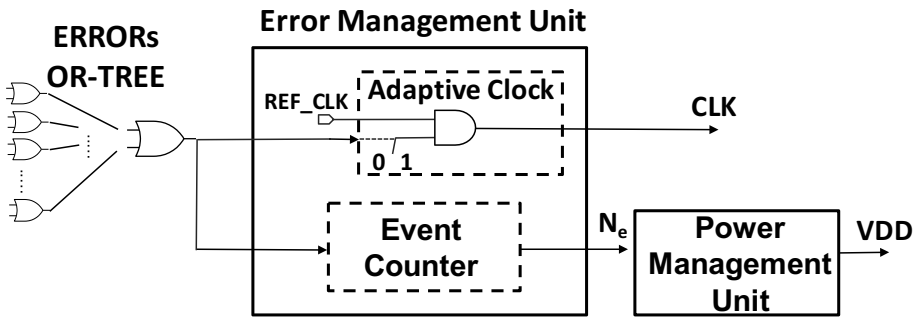


Figure 4.5: Error Management Unit in EBS.

This smart solution allows large detection windows (i.e., $DW = 50\%T_{clk}$) w/o any modification of the clock distribution network. Once detected, the error is locally corrected through logic masking, that is, a MUX switches the output with the complement of the wrong signal stored in the main FF.

In order to let the corrected value propagate toward the fanout logic, the whole circuit has to be stopped for at least one clock cycle. Such an error-driven clock-gating is managed by the *Error Management Unit* (EMU), Figure 4.5, that uses a superset of the error flags (OR among all the Razor-LM in the circuit) as a clock enable. The EMU is also in charge of collecting the error statistics, i.e., the number of error occurrences N_e within a predefined monitoring period of N clock cycles. The *Power Management Unit* (PMU) uses this feedback to implement the dynamic voltage scaling.

Design Flow

The EBS design flow encompasses three different stages we integrated into a commercial design platform (the *Synopsys*® Galaxy) using wrappers written in TCL:

1. **Logic Synthesis**: a classical timing-driven, low-power logic synthesis run using 28 nm industrial technology libraries characterized at the nominal Vdd=1.10 V.

2. **Identification of critical end-points**: after the clock-tree synthesis, the end-points whose worst-case arrival time at minimum voltage Vdd=0.60 V (lower bound of the voltage scaling range) miss the clock-period T_{clk} are labeled as “critical”.

3. **Razor-LM re-placement and TDL insertion**: for each critical end-point, the standard FF is replaced with a Razor-LM and the TDL properly inserted; the error OR-tree is also synthesized.

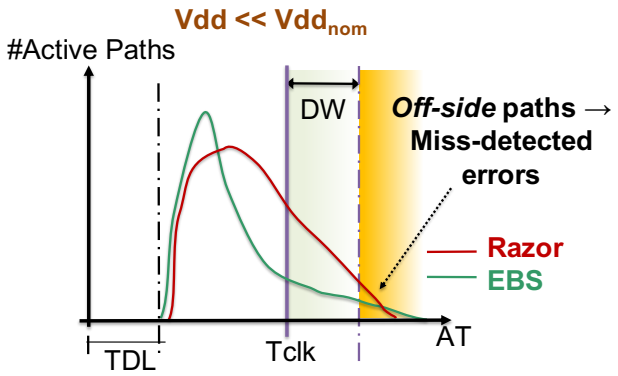


Figure 4.6: Miss-detected errors representation.

4.1.4 On the Efficiency of Early Bird Sampling

AVOS Policies

Adaptive Voltage Over-scaling (AVOS) belongs to the class of adaptive voltage scaling [76] techniques. It implements a context-driven voltage lowering, that is, voltage gets regulated by the occurrence of timing errors on the actual *sensitized* critical paths, i.e., those activated by the actual input patterns.

As already discussed, this may lead some of the longest paths beyond the clock period. Those which fall within the detection window (DW) are detected and eventually corrected; that is the basic principle of EBS. However, there might be specific sequences of input patterns that push the supply voltage so down that some of the longest paths could even exceed the DW; such *off-side* paths represent the primary source of error *miss-prediction*. The latter case is graphically depicted in Figure 4.6. Paths in *off-side* run out of control, and their activation is the main source of error propagation. Here is why AVOS is particularly suited for error-resilient applications.

It is worth to emphasize that miss-detections mainly raise depending on the voltage scaling policy adopted. We, therefore, provide a parametric analysis among different AVOS parameters and different management policies (the latter being described in the next subsection).

The main feedback provided by the error management unit (EMU) is the number of errors N_e within a predefined number of clock-cycles N, the monitoring period (please refer to Section 4.1.3). The power management unit (PMU) makes use of such error-rate ER in order to implement some voltage scaling policy. More specifically, the ER is compared against a given error-threshold ER_{th} (or multiple error-thresholds) in order to trigger the voltage scaling. In this section, we implemented three different policies as follows.

1. Single-threshold (STh): as shown in Figure 4.7 (left), given ER_{th} as a user-defined error threshold, the policy works as follow:

- as soon as N_e gets larger than ER_{th}, the supply voltage is increased w/o waiting for the end of monitoring period.
- if $N_e \leq ER_{th}$ at the end of the monitoring period, i.e., after N cycles, the supply voltage is reduced for power minimization.

To notice that STh enables the control over the minimum Operation per Clock-cycle (OPC), a measure of performance overhead due to error correction; indeed, ER_{th} represents the maximum OPC loss.

2. Double-threshold (DTh): conceived to be more conservative, the DTh policy exploits a "neutral" region defined by two thresholds $ER_{th_{min}}$ and $ER_{th_{max}}$, Figure 4.7 (right); within this region, the supply voltage is kept untouched. This avoids excessive Vdd ripples thus making the voltage scaling smoother. The policy works as follows:

- as soon as $N_e \geq ER_{th_{max}}$, Vdd is scaled up w/o waiting for the end of monitoring period;
- if $N_e \leq ER_{th_{min}}$ at the end of the monitoring period, Vdd is scaled down for power minimization;
- if $ER_{th_{min}} < N_e < ER_{th_{max}}$ at the end of the monitoring period, Vdd is kept unchanged in order to avoid excessive Vdd ripple.

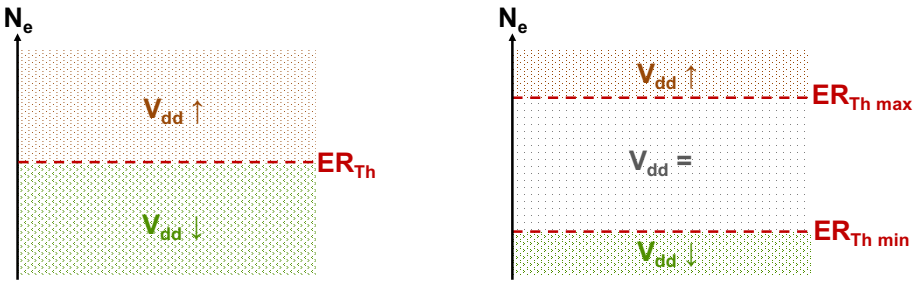


Figure 4.7: STh (left) vs. DTh (right) Vdd scaling policies.

3. Threshold-exceeding Saturation Counter (SC): this policy is more elaborated as it takes into account how the supply voltage evolves over time. The working mechanism, depicted in Figure 4.8 as a Mealy Finite State Machine (FSM), makes use of a 4-bit saturation counter to decide whether the voltage has to be scaled up/down. It works as follows:

- the policy starts reducing Vdd (*Safe* state);
- if N_e exceeds the ER_{th} a warning signal is raised (E=1) at the end of the monitoring period and the FSM state evolves to *Saturation Count*. Vdd is not increased;

- the Vdd is scaled up iif the number of consecutive warning signal c is equal to c_{max} (2^5-1). In this case, the FSM moves to the *Unsafe* state;
- If the current state is *Unsafe* and no warning signal is raised (E=0), the FSM evolves in *Safe* state and Vdd is scaled down;
- anytime FSM reaches the *Saturation Count* state, the warning count c is set to zero.

To notice that the SC policy has been thought to be more aggressive than STh; indeed it allows to increase the time spent at lower Vdd, even when the N_e exceeds ER_{th}. This enables larger energy savings at the cost of some performance and quality-of-results loss.

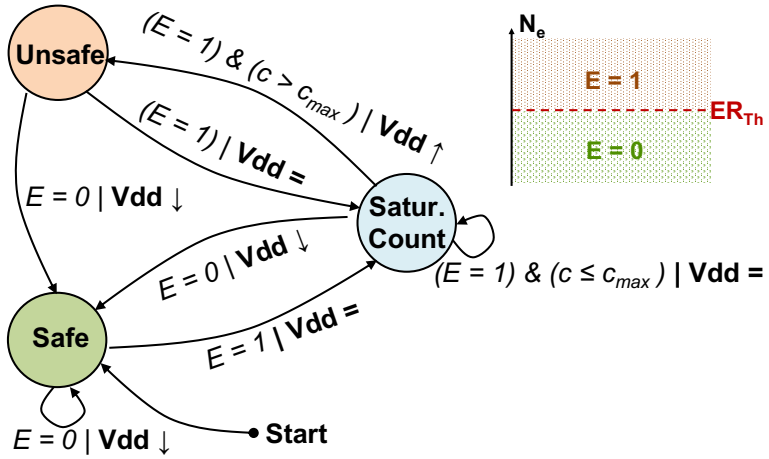


Figure 4.8: Saturation Counter Vdd scaling policy.

Experimental Framework: Benchmarks and Testbenches

The proposed EBS technique has been tested on a set of open source benchmarks over which we applied a AVOS scheme. The five circuits under analysis are:

- *Adder*: 32×32-bit + Carry-In Adder; f_{clk} = 750 MHz.
- *Multiplier*: 32×32-bit Multiplier; f_{clk} = 500 MHz.
- *MAC*: 16×16-bit Multiply Accum. Unit; f_{clk} = 650 MHz.
- *FIR Filter*: Pipelined 16th-order low-pass FIR filter in direct form (12-bit in, 24-bit out); f_{clk} = 650 MHz.
- *IIR Filter*: Pipelined 8th-order low-pass IIR filter in direct form I, modeled after a Bessel analog filter (16-bit in, 32-bit out); f_{clk} = 650 MHz.

For each benchmark, both the EBS and the Razor versions have been designed. The difference between them is the method adopted to solve the short-path races, i.e., TDLs for EBS and standard post-synthesis short-path padding for Razor; in both cases, the number of monitored end-points is the same. The short-path padding procedure implemented for the Razor circuits uses multi-V_{th} clock buffers that minimize the area overheads.

The AVOS is emulated using an in-house tool which runs functional simulations (*Mentor QuestaSim*) with back-annotated `sdf` delay information. Propagation delays are extracted using a Static Timing Analysis engine (*Synopsys PrimeTime*) loaded with technology libraries characterized at different supply voltages; for those supply voltages not available in the library set we used derating factors embedded into the STA. The power dissipation is calculated using probabilistic models (*Synopsys PrimePower*) with back-annotated signal statistics from `saif` format files. The energy consumption is estimated considering the supply voltage profiles collected from simulations.

The emulated workload consists of realistic input stimuli made up of $5\text{x}10^6$ patterns customized for each benchmark. For arithmetic circuits (*Adder*, *Multiplier* and *MAC*) we organized the patterns as sequence of Gaussian distributions each of them having a variable mean; for *Adder* and *Multiplier*: $\mu_1 = 2^8$, $\mu_2 = 2^{16}$ $\mu_3 = 2^{28}$ with standard deviation $\sigma = 2^8$; for *MAC*: $\mu_1 = 2^4$, $\mu_2 = 2^8$ $\mu_3 = 2^{12}$ with standard deviation $\sigma = 2^4$. For *FIR* and *IIR filters*, stimulus consists of a set of baseband audio samples.

For the sake of clarity, plots and bar graphs of experimental results which refers to Razor report the caption **RZ-BF** to indicate that conventional Razor short-path padding has been implemented by *buffers insertion*.

Quality Metrics

1. *Average Vdd*: average of the Vdd measured over the testbench trace.
2. *Energy per Operation* (EPO): ratio between energy consumed and number of operations.
3. *Operation per Clock Cycle* (OPC): ratio between the number of operation run and total number of clock cycles.
4. *Uncovered Errors* (UE): the count of logic errors due to undetected timing faults occurred during simulation. This metric is measured in *ppm* (parts per million).
5. *Normalized Root Mean Squared Error* (NRMSE):

$$NRMSE = \sqrt{\frac{\sum_{i=0}^{n}(y[i] - y_o[i])^2}{n}} \cdot \frac{1}{y_{max} - y_{min}} \tag{4.2}$$

with y the value sampled at the output of the circuit, y_o the right output value, n is the total number of operations; y_{max} and y_{min} are the max and the min value of y_o, they define output dynamic. $NRMSE$ quantifies the QoR.

To be noticed that our simulations do not consider process variations as they do not affect the functionality of the proposed technique.

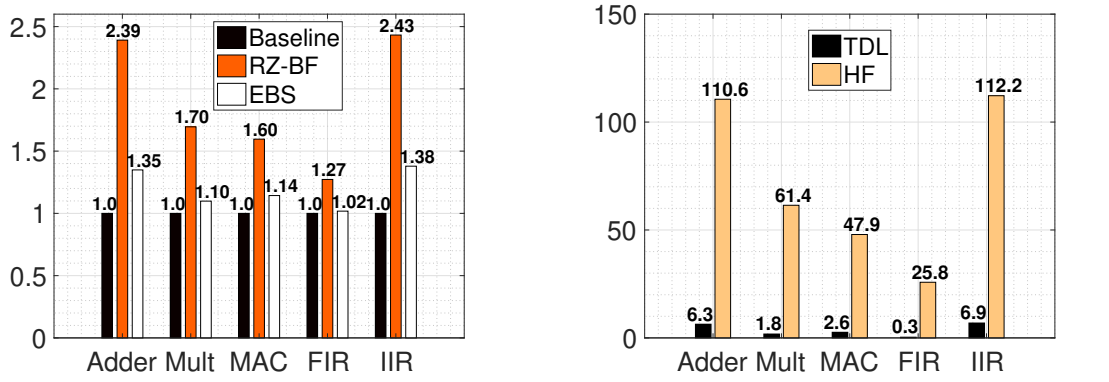
Area Overhead

Table 4.1 collects the statistics of the five benchmarks; column **#FFs** reports the total number of flip-flops (FFs), while column **#Critical-FFs** the percentage of FFs replaced with timing monitors, the Razor-LM.

Benchmark	Area [μm^2]	#FFs	#Critical-FFs	DW [ps]	TDL [ps]
Adder	339.45	98	22.4%	665	616
Mult	2954.01	128	42.2%	1000	898
MAC	1241.12	72	45.8%	750	656
FIR	1946.32	228	8.3%	750	634
IIR	3296.80	296	78.4%	750	692

Table 4.1: Benchmarks designed for EBS.

The DW is set to 50%·T_{clk}, while *TDL* is sized according to equation 4.1. The analysis reported in [165] ensures that the circuit adopted to implement the delay lines (Section 4.1.3) allows to achieve the values reported in the Table 4.1.



(a) Total Area Overhead

(b) Buffers Hold-Fixing vs. TDL area overhead (%)

Figure 4.9: Area Overhead Comparison.

The bar-charts in Figure 4.9a and Figure 4.9b provide a more detailed area comparison between EBS and Razor; both the implementations show the same number of timing monitors (Table 4.1). Figure 4.9a shows the overall area overhead normalized w.r.t. the baseline circuit (i.e., w/o any EDC scheme). EBS is by far a more compact architecture; average area overhead is 19.8% against 87.8% of Razor. That's due to the efficiency of the

proposed dynamic short-path padding. As shown in Figure 4.9b, TDLs requires much less area(3.6% on average) than buffers insertion using short-path padding procedures (71.6% on average). For instance, the *IIR*, which shows a large number of short-paths in the feedback network, the area penalty of Razor is 112.2%, while it drastically reduces to 6.9% with EBS.

AVOS Improvement with EBS

In order to quantify the improvements brought by EBS, Table 4.2 and Table 4.3 summarize the results achieved during AVOS emulation on the five benchmarks under analysis. The two tables report a collection of the quality metrics presented in Section 4.1.4. Collected results refer to the *single-threshold* policy (STh) described in Section 4.1.4 assuming (*i*) a monitoring period N=10^3 clock cycles, (*ii*) a Vdd step 20 mV, (*iii*) two different values for the error-rate ER_{th}: 2%, i.e., 20 errors in 10^3 cycles - Table 4.2), and 5%, i.e., 50 errors in 10^3 cycles - Table 4.3.

A large ER_{th} accelerates the voltage scaling, hence, it may induce some performance penalty (due to more errors to be corrected) and some QoR degradation (due to a possible increase of miss-detected errors).

Except for the *Mult* benchmark, which we discuss later as a special testcase, the results clearly show EBS outperforms Razor. The savings achieved with EBS are quantified by (*i*) the average Vdd recorded during testbench simulations (column Vdd_{avg}), and (*ii*) the energy-per-operation savings w.r.t. the baseline circuit (column $\text{EPO}_{savings}$). The EBS implementation reaches lower Vdd_{avg} (and also minimum Vdd - column Vdd_{min}) for both the ER_{th} thresholds. This translates into larger EPO savings w.r.t. Razor. Best cases have been measured for *MAC* (48.6% for EBS vs. 9.6% for Razor at ER_{th}=5%) and *Adder* (44.4% for EBS vs 20.7% for Razor at ER_{th}=5%). It is worth to emphasize that in the worst-case (*FIR*), EPO savings achieved with EBS are 2.2× larger than those obtained by Razor: 28.2% vs. 12.8% for ER_{th}=2%; 34.0% vs. 21.5% for ER_{th}=5%.

The *IIR filter* is a kind of circuit for which Razor results quite inefficient; the EPO increases w.r.t. the baseline circuit leading to negative savings: -47.7% at ER_{th}=2% and -45.6% at ER_{th}=5%. Such a huge design overhead is due to the fact that short-path padding overwhelms the power savings of voltage scaling. By contrast, EBS still gets remarkable EPO savings: 30.6% at ER_{th}=2% and 37.6% at ER_{th}=5%.

For what concerns performance degradations due to errors correction, Tables 4.2 and 4.3 clearly shows EBS guarantees a OPC close to that of the Razor strategy: $OPC \geq \{0.98, 0.95\}$ for both the thresholds $ER_{th} = \{2\%, 5\%\}$. This confirms once again TDLs insertion has a marginal effect on the error-rate.

Remarkable results have been also observed in terms of reliability. Although EBS pushes Vdd to values below those achieved with Razor, the number of miss-detections UE is zero for all the benchmarks. The two exceptions are *Adder* (UE=6 ppm and UE=15 ppm, with ER_{th} equals to 2% and 5% respectively) and *MAC* (UE=6 ppm at ER_{th}=5%). Nonetheless only marginal QoR degradation has been observed: $NRMSE$

Benchmarks	EBS					
	Vdd_{min} [V]	Vdd_{avg} [V]	EPO savings [%]	OPC	UE [ppm]	NRMSE [%]
MAC	0.74	0.87	43.6	0.98	0	0
Adder	0.60	0.83	41.9	0.98	6	0.001
IIR	0.92	0.95	30.6	0.98	0	0
FIR	0.84	0.96	28.2	0.98	0	0
Mult	1.10	1.10	-5.5	0.98	0	0
Benchmarks	**Razor**					
	Vdd_{min} [V]	Vdd_{avg} [V]	EPO savings [%]	OPC	UE [ppm]	NRMSE [%]
MAC	0.94	1.00	9.0	0.98	0	0.0
Adder	0.66	0.87	18.6	0.99	35	0.001
IIR	1.00	1.02	-47.7	0.97	0	0
FIR	0.86	0.98	12.8	0.98	0	0
Mult	1.00	1.04	-23.7	0.98	0	0

Table 4.2: Results summary for AVOS set as $N = 10^3$ (clock cycles) and $ER_{th} = 2\%$. Notes: EPO savings w.r.t. Baseline.

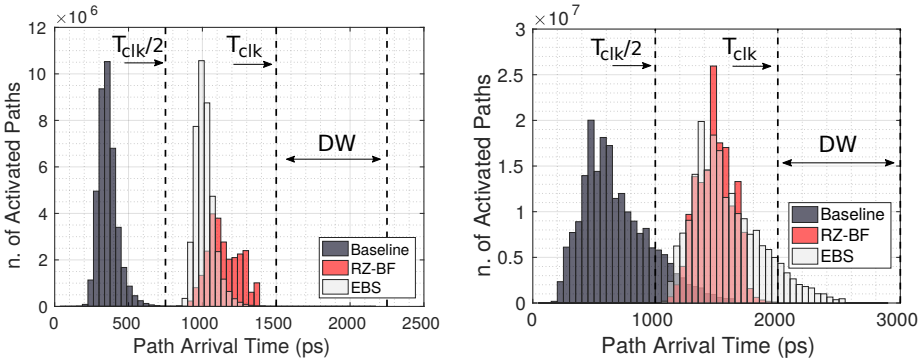
Benchmarks	EBS					
	Vdd_{min} [V]	Vdd_{avg} [V]	EPO savings [%]	OPC	UE [ppm]	NRMSE [%]
MAC	0.72	0.83	48.6	0.95	6	0.128
Adder	0.60	0.81	44.4	0.97	15	0.001
IIR	0.88	0.90	37.6	0.95	0	0
FIR	0.80	0.93	34.0	0.95	0	0
Mult	1.10	1.10	-8.5	0.95	0	0
Benchmarks	**Razor**					
	Vdd_{min} [V]	Vdd_{avg} [V]	EPO savings [%]	OPC	UE [ppm]	NRMSE [%]
MAC	0.94	0.99	9.6	0.95	0	0
Adder	0.66	0.86	20.7	0.97	91	0.002
IIR	0.98	1.01	-45.6	0.96	0	0
FIR	0.86	0.96	21.5	0.96	0	0
Mult	1.00	1.02	-19.5	0.96	0	0

Table 4.3: Results summary for AVOS set as $N = 10^3$ (clock cycles) and $ER_{th} = 5\%$. Notes: EPO savings w.r.t. Baseline.

is a mere 0.128% at worst case. Such a low QoR degradation is achieved thanks to the internal logic topology of the circuits which, in turn, reflects into a low activation of the most critical paths.

As a counterexample, the *Mult* benchmark belongs to that class of circuits whose internal characteristics are not particularly suited for aggressive voltage over-scaling. Both EBS and Razor fail, suggesting AVOS might not be a valuable low-power option. To better understand the reasons behind such behavior, we resort to a comparison between two benchmarks, the *Mult* (for which AVOS does not work) and the *MAC* (for which AVOS gets substantial savings). Figure 4.10 recalls the qualitative analysis discussed in Section 4.1.2. More specifically, it shows the dynamic path distribution of three different implementations: baseline, EBS, and Razor. The bars represent the cumulative number of timing path activations vs. their arrival time.

Some key comments are as follows. First. For both EBS and Razor the path distribution is skewed such that none of the short-paths falls behind $T_{clk}/2$ (the width of



(a) 16×16 bit MAC (b) 32×32 bit Multilplier

Figure 4.10: Dynamic Path Distribution analysis.

the detection window *DW*). This avoids short-path races thus ensuring the right functionality of the error detection mechanism. Second. EBS keeps the path distribution unchanged (just a right shift of the baseline distribution) avoiding the growth of those "quasi-critical" paths that, just standing behind T_{clk}, may prevent voltage lowering. Usually, Razor works on the opposite direction instead, as the number of "quasi-critical" increases due to timing-constrained buffer insertion. This behavior is quite evident for *MAC* (4.10a), for which the red bars (Razor implementation) stand over the white ones (EBS implementation). Since a larger number of activated "quasi-critical" paths reduce the chance of Vdd lowering, EBS results to be more efficient. That is what makes EBS outperforming Razor. However, there might be particular circuits for which this feature does not hold. Such circuits are those for which the basic principle under which EBS is built, namely, *the longer the path, the lower its activation*, gets weaker. That is the *Mult*. As reported in Figure 4.10b, the original dynamic path distribution (baseline implementation) is pretty large, with very active paths that take the whole clock-period. This negatively affects EBS, where the TDLs push many paths into the DW; as a result, the supply voltage is stuck at high values and the EPO gets larger than the original circuit due to error corrections: 1.06× and 1.09× for ER_{th}=2% and ER_{th}=5% respectively. Also Razor suffers from the same problem, as the number of active paths across T_{clk} is huge; EPO increases w.r.t. the baseline circuit: 1.24× and 1.20× for ER_{th}=2% and ER_{th}=5% respectively. However, the overhead of Razor gets larger than that of EBS.

As a final comment, one should consider that circuits on which AVOS does not work properly, may radically change their behavior when integrated into more complex architectures. That is the case of *Mult* integrated into *MAC*.

EBS Characterization Under Different AVOS Implementations

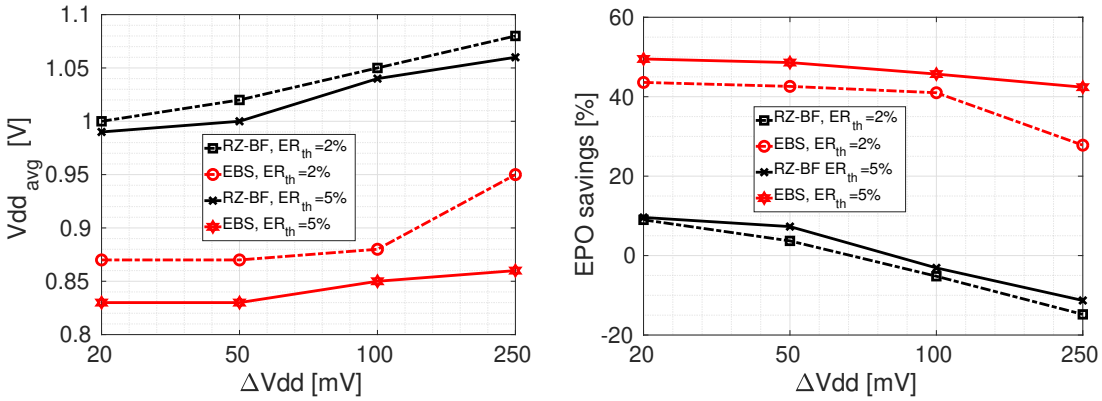
The main goal of this section is to quantify the figures of merit of EBS under different AVOS settings, and thus, to demonstrate EBS performs well under several power

management scenarios. We therefore characterize the quality metrics according to: *(i)* the Vdd step, namely, the ΔVdd used for voltage scaling; *(ii)* the monitoring period, that is, the clock cycles N used to measure the error-rate; *(iii)* the Vdd scaling policies presented in Section 4.1.4. For the sake of space, we just report the analysis for *MAC*. Similar results hold for the other benchmarks.

1. Vdd step

The collected results refer to three different values of ΔVdd: 20 mV, 50 mV, 100 mV, 250 mV. In order to make the analysis more realistic, we also take into consideration different voltage steps may require different clock-cycles to be properly delivered; we therefore assume a latency of {1, 2, 5, 12} clock cycles for {20 mV, 50 mV, 100 mV, 250 mV} respectively.

Simulations are conducted on EBS and Razor using the *Single Threshold* Vdd scaling policy (STh) under two different values of error-threshold, ER_{th}=2% and 5%.

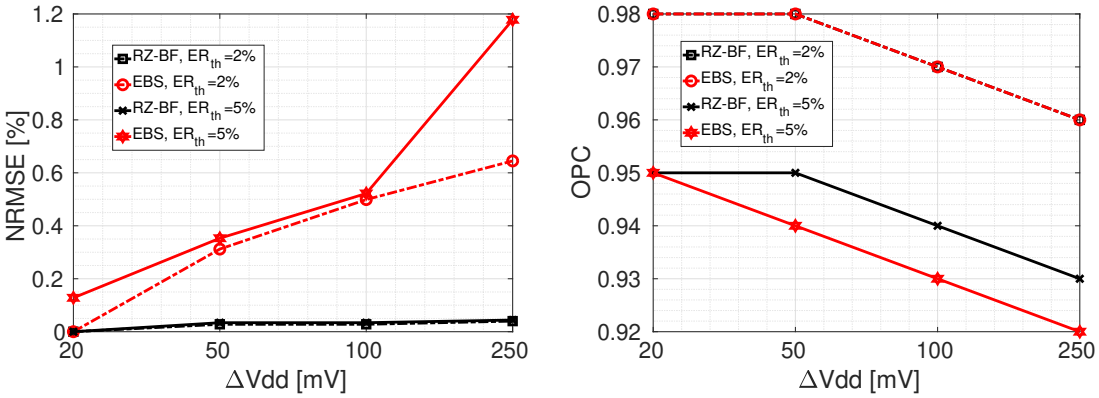


(a) Vdd_{avg} (b) EPO savings (w.r.t. Baseline)

Figure 4.11: Energy efficiency vs. Vdd Step width (ΔVdd).

EBS outperforms Razor for any of the Vdd-steps values under analysis. As shown in Figure 4.11a, EBS brings the circuit to a lower average Vdd; along the whole ΔVdd range, the average improvement w.r.t. Razor reaches 140 mV at ER_{th}=2% and 180 mV at ER_{th}=5%. The same results hold for energy efficiency; average EPO savings are: 46.6% and 38.8% at ER_{th}=5% and ER_{th}=2% for EBS vs. a mere 0.6% and 0.01% for Razor. More in details, Figure 4.11 shows how savings drift with ΔVdd. Vdd_{avg} reaches lower values using a finer voltage resolution. For instance, considering the EBS at ER_{th}=2%, it reduces from 0.95 V at ΔVdd=250 mV to 0.87 mV at ΔVdd=20 mV. As a result, the energy savings reported in Figure 4.11b show substantial improvements, from 27.8% at ΔVdd=250 mV to 43.6% at ΔVdd=20 mV.

The voltage resolution does also impact the QoR. As shown in Figure 4.12a, the lower the Vdd step, the better the QoR. Indeed, a larger ΔVdd makes harder to control



(a) $NRMSE$ (%) (b) Operation/Clock-cycle

Figure 4.12: QoR and perfomance vs. Vdd Step width (ΔVdd).

the occurrence of miss-detected errors. For ER_{th}=5%, the $NRMSE$ measured for EBS reduces from 1.2% at ΔVdd=250 mV to 0.1% at ΔVdd=20 mV. By contrast, the $NRMSE$ of the Razor implementation is less sensible (variation in the range [0.0% - 0.1%]); that is mainly due to the fact that the Vdd scaling is slower than in EBS, therefore, fewer miss-detections do occur.

Concerning the performance, the *STh* Vdd-scaling policy is conceived as a mechanism to control the minimum OPC value; ideally, as introduced in Section 4.1.4, ER_{th} represents the max. OPC loss, thus the minimum OPC equals $1 - ER_{th}$. However, since larger Vdd steps come with larger latencies, the performance achieved by EBS and Razor are substantially affected, and OPC may drop below that ideal minimum boundary if OPC loss $>$ ER_{th}. Figure 4.12b shows this drawback through OPC vs. ΔVdd plot; both EBS and Razor OPC losses are still kept lower than ER_{th} only for ΔVdd=20 mV (both the ER_{th}s). In the worst case, i.e., ER_{th}=5%, OPC loss raises from 5% (i.e., the ideal max. loss value) to 8% for EBS and from 5% to 7% for Razor in the interval ΔVdd=[20 mV - 250 mV].

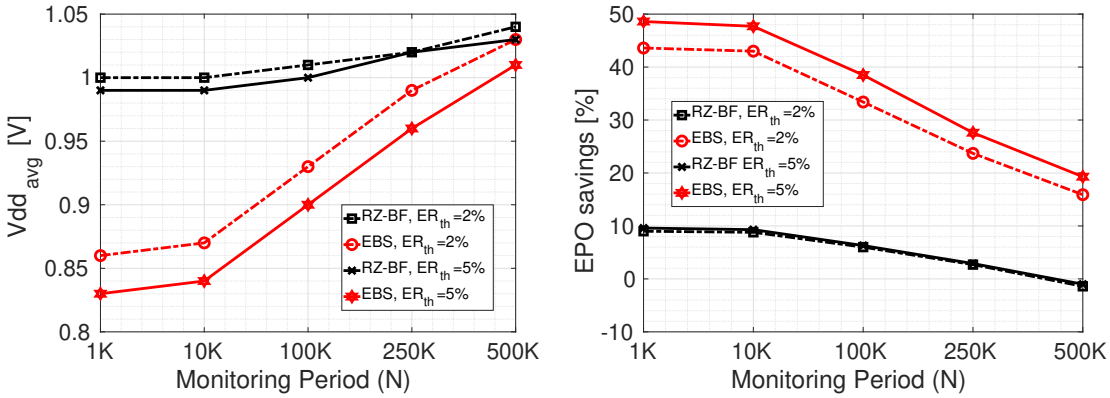
2. Monitoring Period

The plots reported in Figure 4.13 show Vdd_{avg} and EPO using different monitoring period N. Simulations are conducted on EBS and Razor using the Single Threshold Vdd (STh) scaling policy ΔVdd=20 mV and two different values of error-threshold, ER_{th}=2% and 5%.

EBS performs more efficiently than Razor for all the operating conditions. Considering the case ER_{th}=5% (the best case), EBS reaches lower Vdd_{avg}, 0.91 V vs 1.02 V of Razor (Figure 4.13a), and larger EPO savings, 36.4% vs. 5.4% of Razor ((Figure 4.13b), average values on N interval).

As a general rule, the larger the N, the slower the Vdd scaling. While this trend is

less evident in Razor (Vdd_{avg} increases by just 40 mV), EBS amplifies the effect showing an overall spread of 180 mV (from 0.83 mV to 1.01 mV). The same consideration can be inferred for EPO, where savings gets smaller with N, from 48.6% (N = 10^3) to 19.3% (N = $5{\cdot}10^5$).



(a) Vdd_{avg} (b) EPO savings (w.r.t. Baseline)

Figure 4.13: Energy efficiency vs. Monitoring Period (N).

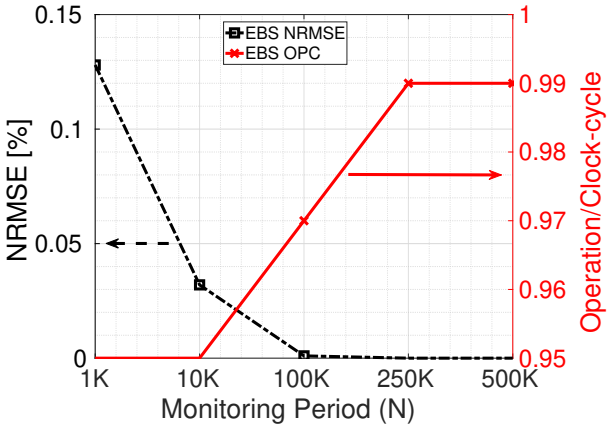


Figure 4.14: EBS with ER_{th}=5%: QoR and performance vs. Monitoring Period (N).

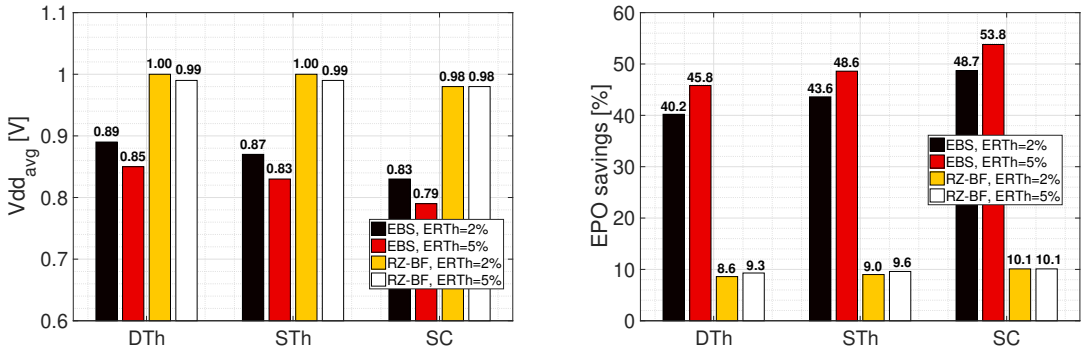
Finally, the analysis reported in Figure 4.14 shows that a more aggressive voltage scaling strategy, i.e., smaller N, affects output quality and performance due to an increasing number of error corrections. Results for Razor are omitted as the $NRMSE$ always gets zero whatever the value of N. The $NRMSE$ of EBS increases, yet, only marginally: from zero to 0.128%. Also the OPC drops: from 0.99 to 0.95 when N reduces from =$5{\cdot}10^5$ to 10^3 for both EBS and Razor. That's the cost to be payed for a more energy efficient AVOS.

3. AVOS Policies

Simulations of the three Vdd scaling policies described in Section 4.1.4 have been run fixing the AVOS parameters as follows:

- Vdd step, ΔVdd=20 mV;
- monitoring period, $N = 10^3$.
- error-threshold: ER_{th}={2%, 5%} for *STh* and *SC*, $ER_{th_{max}}$=ER_{th}={2%, 5%} and $ER_{th_{min}}$=0.2·$ER_{th_{max}}$ for *DTh*.

Figure 4.15 plots the collected results, which show once again EBS improves the figures of merit of AVOS, whatever the adopted policy.



(a) Vdd_{avg} (b) EPO savings (w.r.t. Baseline)

Figure 4.15: AVOS Policies energy efficiency comparison.

Benchmarks	Razor				EBS			
	OPC		NRMSE		OPC		NRMSE [%]	
	ER_{th}=2%	ER_{th}=5%	ER_{th}=2%	ER_{th}=5%	ER_{th}=2%	ER_{th}=5%	ER_{th}=2%	ER_{th}=5%
Double Th	0.99	0.97	0.000	0.000	0.99	0.97	0.000	0.000
Single Th	0.98	0.95	0.000	0.000	0.98	0.95	0.000	0.128
Saturation Counter	0.87	0.86	0.000	0.000	0.95	0.90	0.158	0.185

Table 4.4: Results summary for AVOS Policies OPC and $NRMSE$.

The *DTh* is more conservative. In this case EBS reaches a lower Vdd_{avg} than that of Razor: at ER_{th}=5% (best case), 0.85 V vs. 0.99 V. With a lower Vdd, also EPO savings improve: 45.8% vs 9.3%. At the opposite corner, the *SC* approach pushes a more aggressive Vdd scaling. EBS reaches the lowest Vdd_{avg}, hence, the largest EPO savings: 0.79 V

with 53.8% energy savings, against 0.98 V and 10.1% of Razor. This comes at the cost of some miss-detection. As shown in Table 4.4, the adoption of the *SC* policy induces a $NRMSE$ degradation, mainly due to miss-detected errors: 0.185% in the worst case (ER_{th}=5%). By contrast DTh ensures zero miss-detected errors, both for EBS and Razor.

For what concerns performance, DTh affects OPC only marginally: 3% loss for both EBS and Razor in the worst case (ER_{th}=5%); DTh reduces Vdd ripples thus bringing to a lower number of error corrections. On the contrary, *SC* heavily impacts performance with OPC loss in the order of 10% for EBS and 14% for Razor (at ER_{th}=5%).

4.2 Tunable Error Detection (TunED)

Early Bird Sampling (EBS) has provided the first improvement toward the design of the *Approximate Error Detection-Correction* (AED-C) error management for Energy-Accuracy scaling, i.e., a lightweight method to implement a dynamic Short-path padding. In this section, the second key feature of AED-C is proposed: a Tunable Error Detection (TunED) mechanism to efficiently drive energy-accuracy scaling [125].

The abstract view of the proposed TunED architecture is reported in Figure 4.16. The standard design of a Razor-FF [39] is enhanced with the insertion of a *Tunable Detection Window* (TDW), i.e., a tunable delay that introduces a variable clock skew between the main FF and the shadow FF. Modifications on the TDW width alter the resolution of the error detection mechanism, and hence the error coverage: *the larger the width of the TDW, the larger the number of errors Razor-FFs can detect, and hence correct.* Variations on the TDW impacts on the accuracy-energy tradeoff. While a higher number of detected errors makes the the voltage scaling slower and the QoR higher, a smaller TDW gives voltage scaling much smaller inertia allowing Vdd to reach lower values. The latter case guarantees more energy savings at the cost of some QoR penalty. That is why TunED is the key feature for implementing AED-C strategies for adaptive EAS. The benefits of TunED mechanism against standard Razor timing sensors are clarified in the next section, as the details of AED-C driving a dual-mode Adaptive Voltage Over-Scaling (AVOS) power management are disclosed.

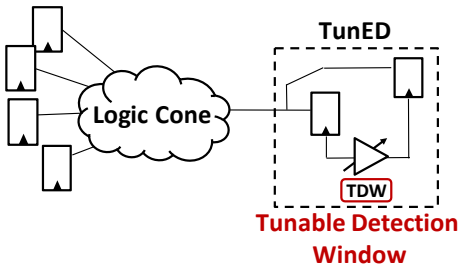


Figure 4.16: TunED implementation abstract view.

4.2.1 TunED Circuit Implementation

The TunED circuitry we propose consists of Razor-FFs [39] augmented with a TDW and a logic masking circuitry for error correction [126], Figure 4.17. When the input of the main FF switches after the rising edge of the clock, i.e., if a timing error occurs, the XOR produces an error flag; the latter is sampled by the shadow latch once the detection window elapses. The MUX locally corrects the error by switching the primary output Q to the complement of the main FF.

Concerning the design of TDW, we borrowed the solution proposed in Section 4.1.3

as possible circuit implementation. It consists of a pair of inverters with a voltage-controlled variable load in between them; the load is a transmission gate whose ON-resistance is controlled by V_{delay}. As discussed in [165], this solution allows a wide range of delays at low area overhead; however, other solutions may work as well.

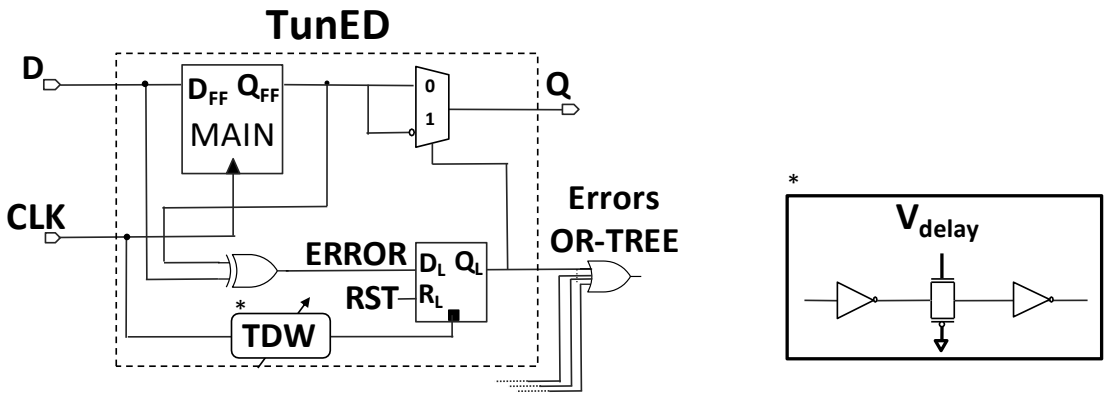


Figure 4.17: TunED circuitry.

4.3 Approximate Error Detection-Correction (AED-C) for Efficient EAS

As explained in Section 4.1, *Early Bird Sampling* exploits the opportunity Adaptive EAS gives: the full error coverage does not need to be ensured. Thus, the large area/energy overhead of conventional EDC methods is reduced by implementing simpler error detection circuitry, i.e., using Tunable Delay Lines instead of heavy chains of buffers to solve short-path races, and reducing the complexity of error recovery mechanism through logic masking circuitry. EBS also introduces a dynamic short-path padding mechanism, a key element for enabling the TunED mechanism, as it is explained later in the text. TunED plays a fundamental role as it relies upon an important principle: the detection coverage capability, precisely the error Detection Window (DW), can be used to drive an efficient energy-accuracy scaling.

Early Bird Sampling (EBS) and *TunED* are the two core ideas on which AED-C is built on, as Figure 4.18 depicts. For the sake of clarity, the following sections first briefly recall all the steps that enable the conversion from Razor to AED-C. Then, the circuit implementation details are explained. The experimental setup, over a set of image/audio processing error-resilient applications, shows AED-C driving a dual-mode AVOS: (i) *Slow* AVOS mode, where DW is taken as large as possible, 50% of the clock period (T_{clk}), such that error detection is maximized; (ii) *Aggressive* AVOS mode, where DW is reduced in order to relax the error detection accuracy and to give AVOS the opportunity to maximize energy savings at the cost of Quality-of-Results (QoR). The power

management unit may select the proper mode depending on the requirements imposed at the application level and/or other external variables, such as the remaining battery lifetime. The choice of employing a dual-mode AVOS power management is crucial as it helps to clarify AED-C pros and cons.

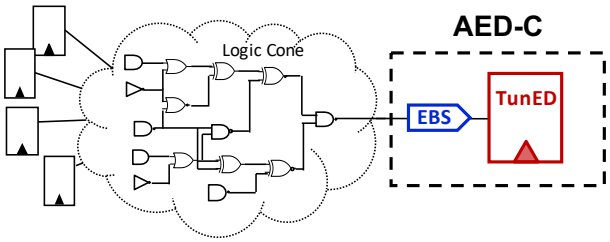


Figure 4.18: AED-C abstract view.

4.3.1 From Razor to AED-C for EAS

In conventional Razor-based error detection and correction strategy, the supply voltage is regulated using the Error Rate (ER), that is the number of timing faults detected during a *monitoring period*. The latter is an integer multiple of the clock-period, while the number of timing faults is the number of clock-cycles during which at least one Razor-FF detected a timing violation. It is worth noting that the error flags generated by each Razor-FFs are all OR-ed, thus generating a global error signal that represents the timing compliance of the whole circuit. If ER is smaller than a user-defined threshold ER_{th}, then the Vdd can be lowered, otherwise Vdd is raised. Different control policies can be implemented that alter the voltage-scaling and ER_{th} is the key parameter to play with: the larger is the ER_{th}, the faster is the Vdd scaling. However, since error correction is a costly procedure, an energy–performance tradeoff exists. A fast voltage lowering may induce latency and power penalties due to the larger number of corrections [38, 31], while a slow voltage lowering is more conservative but it reduces the energy savings. Since there is no general rule, ER_{th} is empirically chosen depending on the design specs. Razor has been conceived to cover the occurrence of *all* the timing faults (*Always-correct* scaling), as shown in Figure 4.19a. This explains why the DW is taken as large as possible (Figure 4.19c), usually 50% of the clock period [38]. To notice that DW is statically defined at design-time and that its implementation hides critical issues that are detailed in the next subsection. The AED-C elaborates on Razor and introduces the concept of Tunable Error-Detection (TunED) [125]. TunED leverages Razor-FF enhanced with a *Tunable Detection Window* (TDW), that is a tunable clock skew between main FF and the shadow FF, Figure 4.19b. TunED can be understood as an elastic Razor-FF. The TDW alters the resolution of the error detection and hence the faults covered by the timing sensors. As graphically depicted in Figure 4.19d, *the smaller the TDW, the larger the number of uncovered latent faults* (hence, Approximate Error Detection). The probability of miss-detected faults gets larger as the TDW reduces. This

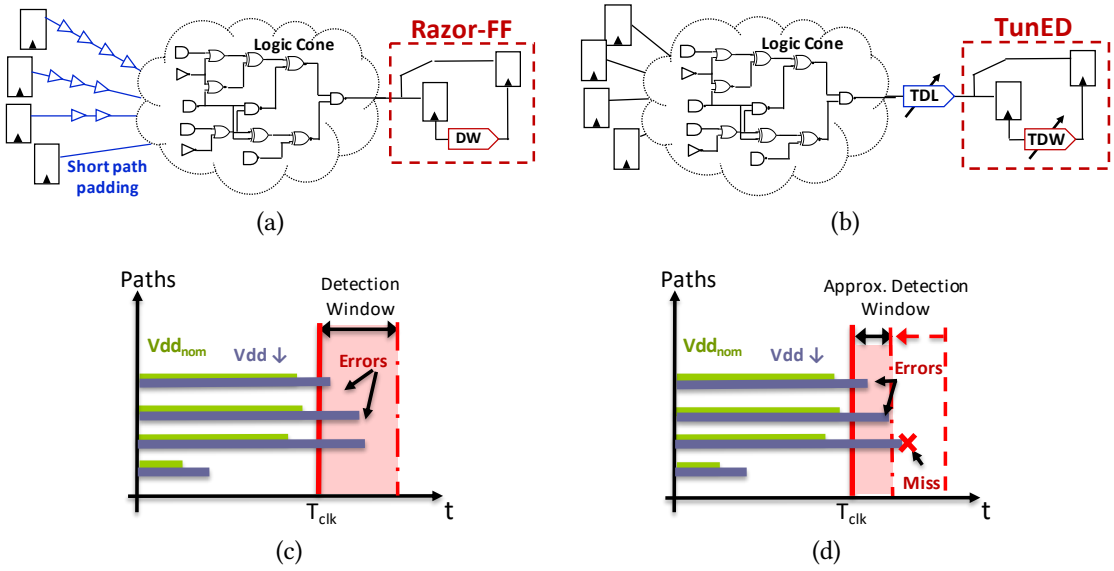


Figure 4.19: Razor (a) vs. AED-C (b) architecture abstract view. *Always-correct* (c) vs. *Approximate Error Detection* (d) EAS.

is the enabling mechanism for AED-C. In fact, the TDW is the knob that regulates the level of approximation in detecting errors. A small TDW implies more miss-detection (lower error rate) and so a faster Vdd scaling which leads the circuit to low energy and low quality, whereas large TDW guarantees high quality, but less energy savings as more faults are properly detected and corrected (higher error rate). When TDW is set as 50% of the clock-period, AED-C approaches the Razor behavior.

Understanding the Dynamic Short-path Padding in AED-C

As explained in Section 4.1, Razor suffers from the so-called *short-path race* which manifests when a *short-path* and a *long-path* connected to the same end-point get sequentially activated at clock-cycle t_i and t_{i-1} respectively. Under such condition, the skewing mechanism implemented within the timing sensors does fail. Razor-FFs, as well as TunED, would not be able to make a distinction between the activation of the short-path within the DW and the activation of the long-path beyond the clock edge. As a result, "false" error detection may occur.

A common practice to address the short-path race is to apply a *short-path padding* using a standard hold-fixing procedure. The latter consists of delaying all the short-paths in a way that their arrival time gets larger than DW. The delay is implemented by means of optimally placed dummy buffers (overview in Figure 4.19a). The resulting effect is qualitatively shown with the left plot of Figure 4.20a, where the solid line is the static path distribution of the original circuit while the dotted line is the same static distribution after the short-path padding. Unfortunately, the short-path padding is a

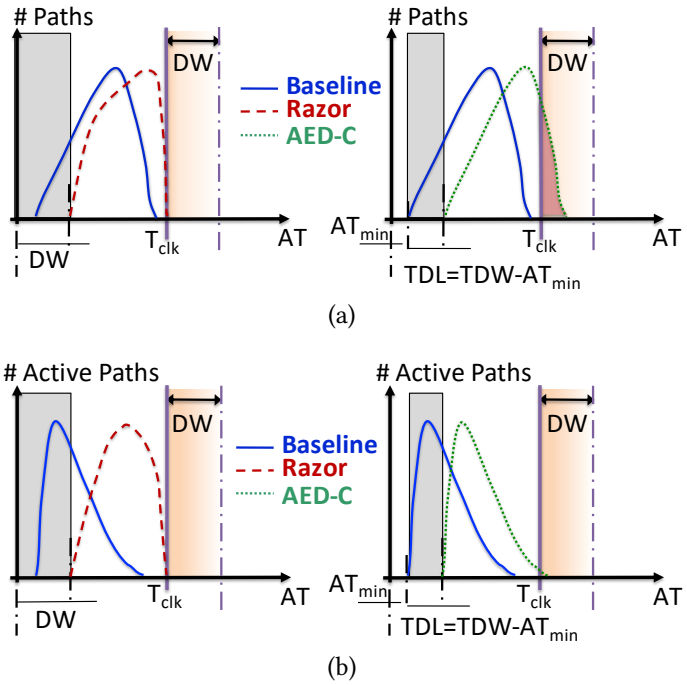


Figure 4.20: Static (a) vs. Dynamic (b) paths distribution: Razor (left) vs. AED-C (right).

static method which does contrast with the dynamic nature of the tunable detection strategy. *Early Birds Sampling* (EBS) (section 4.1) suggests *dynamic short-path padding* as an effective patch. A *Tunable Delay Line* (TDL) is inserted at end-points of the circuit (overview in Figure 4.19b), where timing sensors are placed, then the TDL is adjusted at run-time following the modulation of the detection window. More in detail, the TDL is tuned such that the arrival time of the shortest path (AT_{min}) is delayed beyond the DW. The rule is given as follows:

$$TDL = TDW - AT_{min} \tag{4.3}$$

The effect is qualitatively shown with the right plot of Figure 4.20a, where the solid line is the static path distribution of the original circuit while the dotted line is after the TDL insertion.

As explained in [12], the longest paths have a lower activation probability, and their latent faults rarely get excited, that is the same concept of timing speculation. This is graphically depicted in the right plot of Figure 4.20b, which shows only a very limited subset of frequent paths enter the detection window with negligible effects on the error-rate. Needless to say, much depends on the actual workload. The experiments collected for EBS in Section 4.1 prove this strategy is much more efficient than static short-path padding. Apart from area overhead due to buffers insertion [70] [126] [125], short-path padding also affects the supply-voltage scaling. A compression of the timing paths towards the clock edge (T_{clk}) implies a redistribution of the internal switching activities,

that is, the most active get close to the clock-edge as shown in the left plot of Figure 4.20b. Also known as "wall-of-slack", this issue represents a serious impediment: even small reduction of the Vdd produce a huge number of timing faults as the entire "wall" moves into the detection window of the timing sensors. By contrast, dynamic short-path padding does not suffer from this issue, as the dynamic path distribution grows smoother.

4.3.2 Circuit-level Implementation Details

The circuit architecture of the proposed AED-C consists of a TunED timing sensor augmented with a Tunable Delay Line that enable the EBS dynamic short-path padding. In brief TunED is a Razor-FFs [39] enhanced with a TDW and a logic masking circuitry for error correction (Figure 4.21). A set-up time violation is sensed by a change of polarity at the input of the main flip-flop just after the rising edge of the clock. This event is flagged through a XOR gate that runs a parity check between the signals at pins D_{FF} and Q_{FF}. The error flag is sampled in a shadow latch. Once detected, the error is locally corrected through logic masking: the MUX switches the output Q with the 1's complement of the value stored in the main FF.

Although locally computed, the error correction requires a more complex clock-gating mechanism to guarantee a proper propagation of the new corrected output. This is managed through a dedicated unit called *Error Management Unit* (EMU). The latter implements an error-driven clock-gating where the clock-enable is generated by OR-ing the flag of all the timing-monitors in the circuit. If the timing sensor detects a timing violation, the circuit is halted for one clock-cycle to allow the right propagation of the corrected value (obtained through the logic masking described above). The voltage scaling is operated using the timing error rate. The error flags generated by the timing sensors are OR-ed and the resulting events are counted. Specifically, The EMU is in charge of collecting the error statistics, i.e., the number of error occurrences N_e within a pre-defined monitoring period of N clock cycles. The *Power Management Unit* (PMU) uses N_e as a metric to drive the voltage scaling. As a remark of AED-C pros, N_e, i.e., the timing error rate, can be tuned by playing with the width of the TDW of the TunED sensors.

Although different voltage scaling policies can be implemented depending on the design specs (please refer to section 4.1.4), AED-C has been tested with a single threshold policy which works as follows. Given ER_{th} a user-defined error threshold:

- as soon as $N_e > ER_{th}$, the supply voltage is increased in order to avoid excessive performance penalty due to error corrections;
- if $N_e \leq ER_{th}$ at the end of the monitoring period, i.e., after N cycles, the supply voltage is scaled down for power minimization.

As a recall of the concept explained in Section 3.4.4, ER_{th} has a weak correlation with the arithmetic meaning of the output error. In fact, ER_{th} is a measure of the rate of

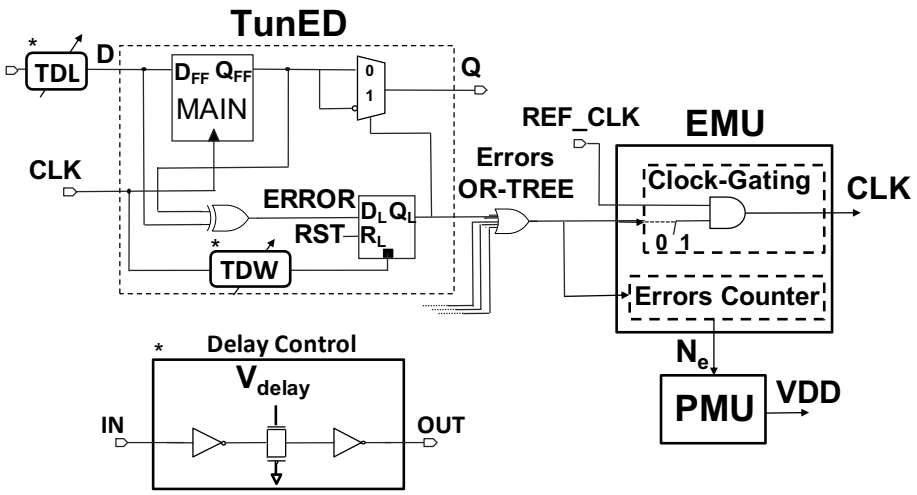


Figure 4.21: AED-C architecture: TunED timing sensor, Error Management Unit (EMU) and Power Management Unit (PMU).

error flags raised by timing sensors which all have the same weights (or "importance"); namely, in AED-C, the quality degradation is bounded by the paths activity. This represents an important distinction when compared to the Algorithm Noise Tolerance (ANT) strategy for example, where $\mathscr{E}_{th}$ error threshold refers directly to the magnitude of the output error.

4.3.3 Experimental Framework: EDA Methods and Tools

AED-C Design Flow

A customized design flow has been integrated into a commercial platform (*Synopsys Galaxy*) by means of dedicated TCL scripts. As reported in Figure 4.22, the flow encompasses three main stages:
1. Logic Synthesis: a classical timing-driven, low-power logic synthesis run using 28 nm industrial technology libraries characterized at the nominal Vdd=1.10V.
2. Critical end-points identification: once the clock-tree has been synthesized, the operating voltage is set to the minimum value Vdd = 0.6 V (lower bound of the voltage scaling range) and the end-points which miss the clock-period T_{clk} are labeled as "critical".
3. AED-C monitors re-placement and TDL insertion: for each critical end-point, standard FFs are replaced with an AED-C monitor and the TDL is inserted; the error OR-tree is also synthesized at this stage. In order to mitigate metastability, the FFs used to implement the AED-C monitor are taken from a *metastability-tolerant* library.

Both TDW and TDL can be tuned at a post-silicon stage, once the nominal T_{clk} is properly adjusted to compensate process-variations. A possible off-line TDW/TDL

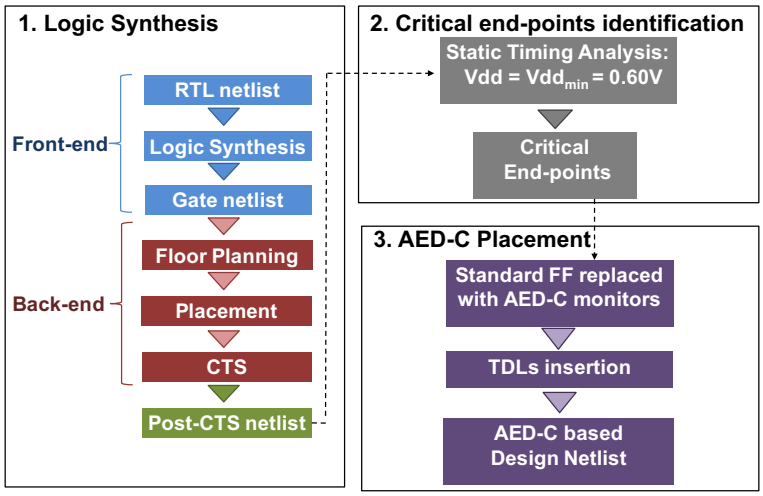


Figure 4.22: AED-C Design Flow diagram.

calibration procedure may be implemented as follows:

(i) The operating voltage is set to Vdd_{nom} and T_{clk} is fixed to the nominal value.

(ii) TDW is fixed to the maximum value (i.e., DW = 50%·T_{clk}).

(iii) The circuit is fed with inputs that excite short-paths and the error rate is measured. As per the internal characteristic of the sensors, the error rate will start very high due to short-path race. The TDL is then increased (by controlling the gate voltage of the transmission gates in Figure 4.21) until error rate reaches zero.

(iv) The circuit is finally tested with a set of input patterns that excite critical paths. If no errors are detected, the calibration procedure stops; otherwise, T_{clk} is increased and the procedure iterates from *(ii)*.

The same calibration procedure is run for both the *Slow* mode and the *Aggressive* mode.

As further remark, AED-C Design Flow could also support the integration of standard low-power design techniques [19].

Benchmarks

The proposed AED-C technique has been tested on a set of open source benchmarks over which we applied a dual-mode AVOS scheme. The three circuits under analysis are:

- *FIR Filter*: pipelined 16th-order low-pass FIR filter in the direct form (12-bit in, 24-bit out); f_{clk} = 650 MHz.
- *IIR Filter*: pipelined 8th-order low-pass IIR filter in direct form I, modeled after a Bessel analog filter (16-bit in, 32-bit out); f_{clk} = 650 MHz.

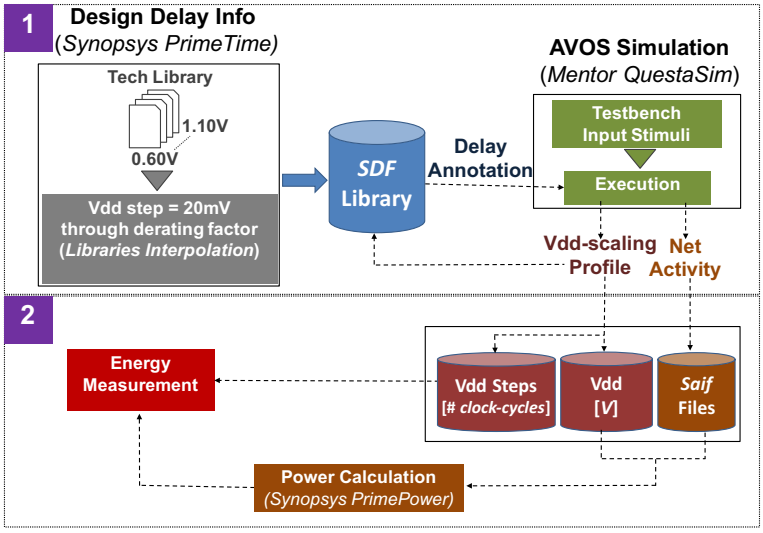


Figure 4.23: In-house AVOS emulation tool diagram.

- *MAC*: 16×16-bit Multiply Accum. Unit; f_{clk} = 650 MHz.

AED-C is benchmarked against a state-of-art Razor AVOS, Razor hereafter. The latter is built with standard Razor-I FFs enhanced with a logic masking circuitry for error correction as in Figure 4.4; the detection window is set to DW=50%·T_{clk}, while the short-path races are fixed through a static short-path padding implemented using a short-path padding procedure that leverages multi-V_{th} clock buffers. For both AED-C and Razor, the number of monitored critical points is the same and the Vdd scaling policy is the one described in section 4.3.2.

AVOS Emulation Tool

An in-house tool that emulates AVOS (Figure 4.23) has been integrated into *Mentor QuestaSim*. It runs a functional simulation with back-annotated SDF delay information extracted through a commercial Static Timing Analysis engine, *Synopsys PrimeTime*, loaded with technology libraries characterized at different supply voltages; for those supply voltages not available in the library set we used derating factors embedded into PrimeTime. The supply voltage ranges from 0.60V to 1.10V with step of 20 mV.

The power dissipation is estimated using a probabilistic power models (*Synopsys PrimePower*) with back-annotated signal statistics extracted from functional simulations using `saif` format files. The energy consumption is estimated considering the supply voltage profiles collected from AVOS emulations.

Quality Metrics

1. Vdd_{avg}: average Vdd obtained during testbench simulation.

2. Vdd_{min}: min Vdd measured over the testbench trace.
3. *Energy per Operation* (EPO): the ratio between energy consumed and number of operations completed.
4. *Operation per Clock Cycle* (OPC): ratio between the number of executed operations and total number of clock cycles.
5. *Uncovered Errors* (UE): the count of logic errors due to undetected timing faults occurred during simulation. This metric is measured in *ppm* (parts per million).
6. *Normalized Root Mean Squared Error* (NRMSE):

$$NRMSE = \sqrt{\frac{\sum_{i=0}^{n}(y[i] - y_o[i])^2}{n}} \cdot \frac{1}{y_{max} - y_{min}} \tag{4.4}$$

with y the value sampled at the output of the circuit, y_o the right output value, n is the total number of operations; y_{max} and y_{min} are the max and the min value of y_o, they define output dynamics. $NRMSE$ quantifies the QoR.

To be noticed that our simulations do not consider process variations as the AED-C is intrinsically "tunable", hence resilient to variability.

4.3.4 Characterization and Figures-of-Merit

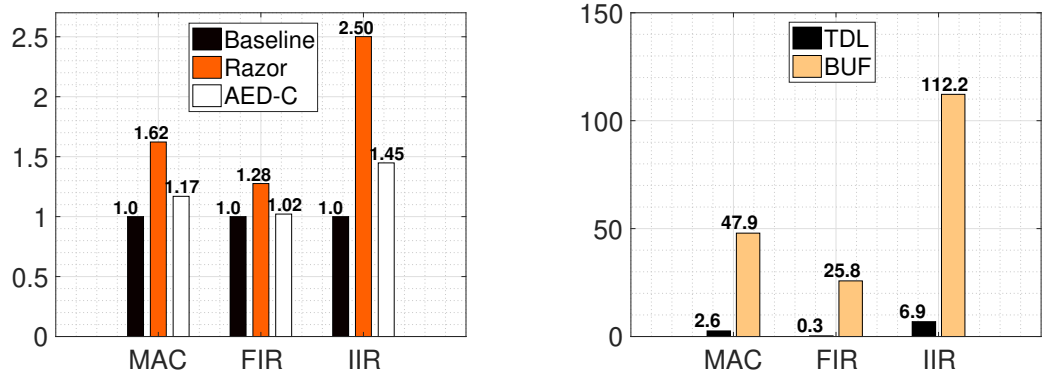
Area Overhead

Table 4.5 collects the statistics of the three benchmarks; column **#FFs** reports the total number of sequential cells, while column **#Critical-FFs** the percentage of timing-critical FFs replaced with timing monitors of Figure 4.21 (logic masking enabled for both AED-C and Razor). The table also reports the value of the *TDLs*, characterized according to equation 4.3, when the TDW is 50% the T_{clk} (750 ps). The analysis reported in [165] ensures that the values of delay in column **TDL** can be achieved with the circuit described in Figure 4.3).

Benchmark	Area [μm^2]	#FFs	#Critical-FFs	TDW [ps]	TDL [ps]
MAC	1241.12	72	45.8%	750	656
FIR	1946.32	228	8.3%	750	634
IIR	3296.80	296	78.4%	750	692

Table 4.5: Benchmarks designed for AED-C.

The bar-charts in Figure 4.24a and Figure 4.24b provide a more detailed area comparison between AED-C and Razor; to note that both implementations have the same



(a) Total area overhead

(b) Buffers vs. TDLs area overhead

Figure 4.24: Area Overhead Comparison.

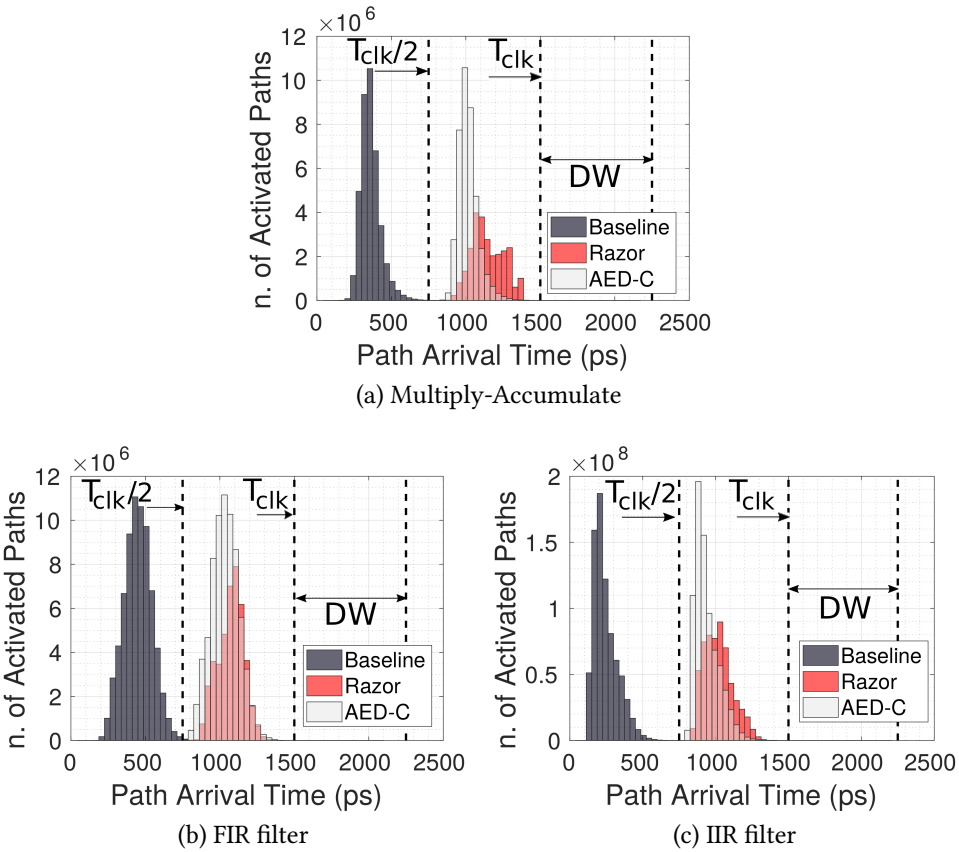
number of timing monitors. Figure 4.24a shows the overall area overhead normalized w.r.t. the baseline circuit. As one can observe, AED-C is by far a more compact architecture; average area overhead is 21.3% against 80.2% of Razor. That's due to the efficiency of the TDL insertion strategy that, as shown in Figure 4.24b, takes much less area (3.3% on average w.r.t. Baseline) than standard buffers insertion procedures (62.0% on average w.r.t. Baseline). Taking as an example the *IIR filter* circuit, buffers induce a huge area penalty of 112.2% (w.r.t. baseline), mainly due to the presence of a large number of short-paths in the feedback network, while the TDL insertion cost is a mere 6.9%.

Dynamic short-path padding Validation

As discussed in Section 4.3.1, AED-C driven dual-mode AVOS is enabled by a Dynamic short-path padding solution. The main goal of this section is to demonstrate how Dynamic short-path padding does not affect the responsiveness of error detection-correction mechanism giving an experimental evidence of the rules of thumb over which it has been implemented.

Figure 4.25 recalls the qualitative analysis explained in Section 4.3.1 showing the dynamic path distribution of three different implementations of the benchmarks (*MAC, FIR and IIR Filter*): Baseline, Razor and AED-C. For each histogram, the bars report the cumulative number of timing path activations vs. their arrival time. The dynamic distributions are sampled at nominal voltage (Vdd_{nom}), i.e. Vdd scaling disabled. For the sake of space, we report only the case in which the DW is fixed to $T_{clk}/2$.

Firstly, both AED-C and Razor are preserved from the short-path races as the path distribution is skewed such that none of the short-paths falls behind $T_{clk}/2$ (the width of the detection window *DW*); this ensures the proper functionality of the error detection mechanism.



(a) Multiply-Accumulate

(b) FIR filter

(c) IIR filter

Figure 4.25: Dynamic short-path padding effects on dynamic Paths distribution.

Secondly, the worst-critical paths are so rarely activated that they are not even visible in the plot; any power management strategy that uses these paths as probe might lead to sub-optimal results.

Thirdly, Dynamic short-path padding keeps the path distribution unchanged as it applies a right shift on the baseline distribution. This avoids the growth of the "quasi-critical" paths that stand just behind T_{clk} and that prevent voltage scaling. By contrast, the insertion of buffers due to short-path padding induce the opposite effect in Razor circuits: paths are skewed towards T_{clk} (red bars) forcing an early stop for Vdd scaling.

AED-C for MAC operations: a Characterization

Through AED-C the error detection resolution can be adjusted at run-time by adjusting the DW; this allows to accelerate or slow-down the voltage scaling depending on the context requirements. The main goal of this section is to prove the validity of this intuition.

The reported parametric analysis aims at showing the responsiveness of AVOS and the resulting output quality when the DW width reduces from 50%·T_{clk} to 15%·T_{clk};

simulations runs on the *MAC* benchmark. The testbench consist of realistic input stimuli made up of 5x10^6 input patterns organized as a sequence of variable-mean Gaussian distributions that model the change of context; we considered three different Gaussian distributions: $\mu_1 = 2^4$, $\mu_2 = 2^8$ $\mu_3 = 2^{12}$, all with standard deviation $\sigma = 2^4$. The input patterns are grouped to perform sequences of 1000 multiply-accumulate operations. The Vdd-scaling policy is the one described in Section 4.3.2 with a monitoring period N=1000 clock cycles and ER_{th} fixed to 2% of error rate.

Figure 4.26 show the collected results. For each value of DW, the testbench is run collecting the average value of Vdd during AVOS simulation; output degradation is measured using the $NRMSE$. The Vdd_{avg} (Vdd_{min}) decreases from 0.87 V (0.74 V) when DW is maximum, up to 0.68 V (0.60 V) when DW = 15%·T$_{clk}$. This trend demonstrates that the smaller the DW, the more aggressive the AVOS. On the other hand, when specific sequences of input patterns push the supply voltage so low that some paths delay could exceed the detection window DW, the quality of the output can be affected by undetected errors. Figure 4.26 shows that the lower the Vdd_{avg}, the larger the output error: the $NRMSE$ for the min. value of DW raises to 1.59%. To be noticed that when DW reaches its max. value, the $NRMSE$ drops to zero; that's the proof AED-C ensures the highest possible QoR when it works in *Slow* AVOS mode.

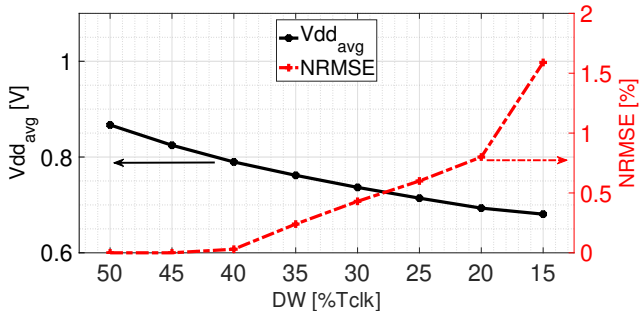


Figure 4.26: TDW characterization.

Table 4.6 reports the EPO savings w.r.t. Baseline circuit (MAC w/o error detection-correction). When AVOS is set to aggressive mode the energy savings increases quickly, from 44.29% at DW= 50%·T$_{clk}$ to 68.55% at DW = 15%·T$_{clk}$.

Concerning performance degradations due to errors correction, OPC does not change show substantial changes w.r.t. DW. The reason lies behind the fact that OPC is mainly affected by the Vdd-scaling policy and the Error Threshold (ER_{th}) adopted (analysis out of the scope of this work).

As further remark, Figure 4.27 reports the Vdd scaling trend during AVOS simulation for the AED-C for max. and min. values of DW (i.e., 50% and 15% of T$_{clk}$), and Razor circuit with DW = 50%·T$_{clk}$. Firstly, it shows that AED-C has a better AVOS profile than Razor: while Razor achieves Vdd_{avg}=1.0 V (EPO savings of 10.1%) AED-C can push the Vdd_{avg} much lower, 0.87 V and 0.68 V for DW equals 50% and 15% of T$_{clk}$, respectively.

DW [%·T_{clk}]	Vdd_{min} [V]	Vdd_{avg} [V]	EPO savings [%]*	OPC	NRMSE [%]
50	0.74	0.87	44.3	0.979	0.00
45	0.72	0.82	50.7	0.979	0.00
40	0.68	0.79	55.7	0.979	0.03
35	0.66	0.76	59.3	0.980	0.24
30	0.64	0.74	62.4	0.980	0.43
25	0.62	0.71	64.9	0.979	0.60
20	0.62	0.69	67.2	0.978	0.80
15	0.60	0.68	68.6	0.981	1.59

Table 4.6: TDW-driven AVOS charazerization on *MAC*
**Note: w.r.t. Baseline.*

Secondly, the AED-C AVOS trends confirm once again the intuition behind this work: smaller DWs (AED-C *Aggressive* mode) accelerate the Vdd scaling, while larger DWs (AED-C *Slow* mode) slow down the Vdd scaling sacrificing energy savings in favor of higher QoR.

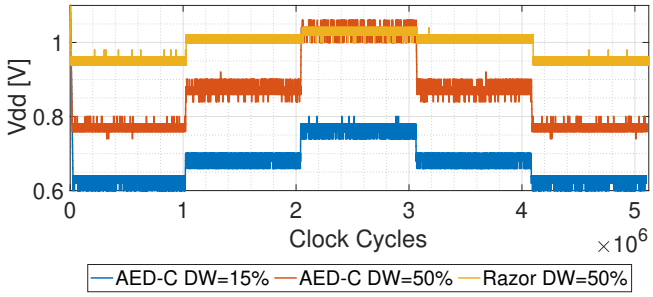


Figure 4.27: Vdd scaling trend vs. DW.

The same trend holds for the other benchmarks under analysis (omitted for the sake of space). We therefore identified the following values as good tradeoff between accuracy drop and energy savings: DW = 50%·T_{clk} for *Slow* AED-C; DW = 25%·T_{clk} for *Aggressive* AED-C.

AED-C for Audio Filtering: a Characterization

In this section, a parametric characterization has been done on spectral audio signal filtering, i.e., FIR and IIR digital filter. They are intrinsically error-resilient and commonly adopted in error-resilient applications. As an additional piece of information, one should consider that the architectures of FIR and IIR substantially differ. The IIR circuit has backward connections which impact the way errors propagate, while the FIR is a feed-forward architecture.

The benchmarks characteristics have been reported in Section 4.3.3: both synthesized and mapped onto a commercial FD-SOI CMOS technology at 28nm at f_{clk} = 650 MHz.

The FIR is a pipelined 16^{th}-order low-pass filter, the IIR is a pipelined 8^{th}-order low-pass filter. The benchmarks were simulated using non-uniform input distributions and realistic input patterns. The simulations aimed at emulating real-life applications where data maintain a certain spatial/temporal correlation, which is the actual condition under which adaptive strategies gain most of their advantage. For this reason, the sequence of three different baseband audio signals was used as test bench. They covered different context scenarios, i.e., different input-data distributions:

- *Audio*-1: Noiseless voice recording; the switching activity of the LSBs is very low for a long portion of the stream.
- *Audio*-2: Taken from an office conversation recording; samples present low noise and inputs have homogeneous switching activity.
- *Audio*-3: Outdoor conversation; the recording is noisy and the switching activity of the inputs quite irregular, due to abrupt changes of input workload.

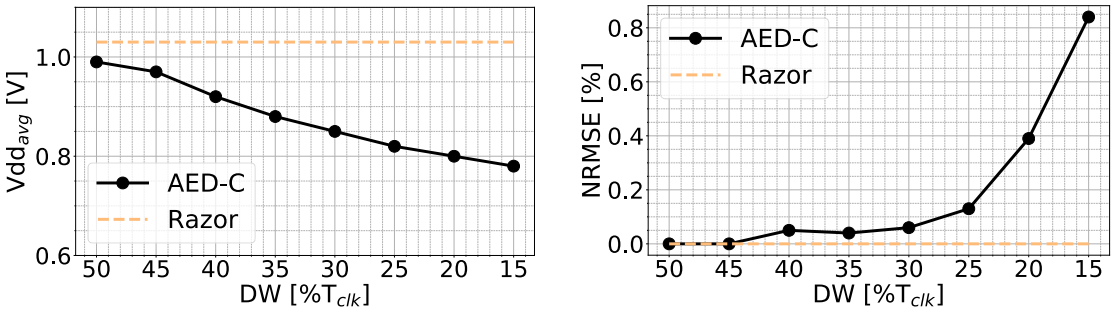
These three different input distributions are very likely to activate a large spectrum of paths, from shorter to longer ones. Interested readers can refer to the work in [125], where a detailed analysis shows how these different input distributions produce significantly different Energy–Accuracy tradeoff.

The proposed parametric analysis is performed against classical Razor to better highlight pros&cons of AED-C. Razor is implemented using the following configuration:

- *Detection Window:* $DW = 50\% \cdot T_{clk}$,
- *Monitoring period:* $N = 10^3$ clock cycles,
- *Error Rate Threshold:* $ER_{th} = 2\%$, in order to limit the performance loss due to errors correction.

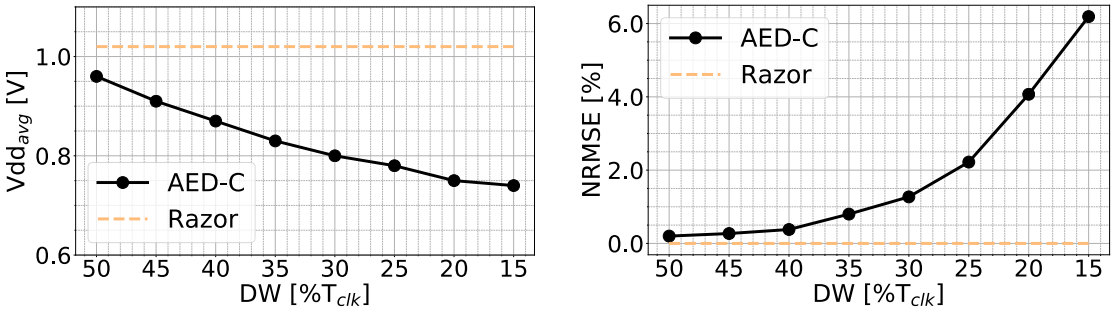
The AED-C is set with the same values, except for the detection window which is used as parameter: $TDW \in [15\% \cdot T_{clk}\ ,\ 50\% \cdot T_{clk}]$, regular intervals of $5\% \cdot T_{clk}$.

Figures 4.28a and 4.29a show the results of voltage scaling efficiency collected for FIR and IIR respectively. While Razor has a fixed DW width, thus a single Vdd_{avg} value highlighted with a dashed line, AED-C enables different Energy-Quality operating points by tuning the DW width. As expected, reducing the DW a more aggressive Vdd scaling can be obtained. In fact, Vdd_{avg} decreases quickly as the DW width become smaller for both FIR and IIR filters. In Razor, the insertion of buffers for short-path padding compress the active paths toward the clock edge inducing an increase of the Error rate. That makes the voltage scaling slower: Vdd_{avg} does not go below 1.03 V (1.02 V) for FIR (IIR). With AED-C the path distribution keeps the same shape (just right shifted) ensuring a smoother increase of the error rate. This is an additional key advantage of the AED-C strategy. The reduction of the error-coverage, namely the reduction



(a) Average Vdd (b) Average Error

Figure 4.28: Razor vs. AED-C characterization: FIR filter.



(a) Average Vdd (b) Average Error

Figure 4.29: Razor vs. AED-C characterization: IIR filter.

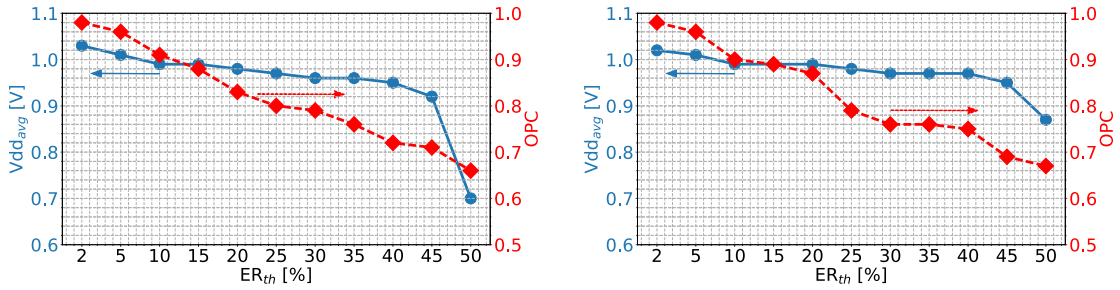
of TDW, implies a proportional reduction of the TDL. As the TDL gets smaller, the path distribution shifts back to its original shape, leaving the most active paths behind the detection window. When considering the FIR benchmark, the Vdd_{avg} reduction is substantial: from 0.99 V at DW=50% $\cdot$ T_{clk}, to 0.78 V at DW=15% $\cdot$ T_{clk}; for the IIR filter case, the Vdd_{avg} reduction is in the range [0.96 V, 0.74 V].

Concerning the quality of the output, while Razor shows zero degradation for both filters, for AED-C the quality of the output reduces as the as the detection mechanism gets more approximated. This trend is clearly reported in Figures 4.28b and 4.29b. The $NRMSE$ increases from 0% at DW=50% $\cdot$ T_{clk} to 0.84% for DW=15% $\cdot$ T_{clk}, while for the IIR filter the $NRMSE$ rises from 0.2% at DW=50% $\cdot$ T_{clk} to 6.19% at DW=15% $\cdot$ T_{clk}. It is worth emphasizing the error shows bigger magnitude since miss-detected timing error traps in the filter feedback network propagating back to the internal paths and persisting until a sequence of input patterns mask it. To be noticed that the IIR filter implemented with AED-C presents an average error greater than zero when DW=50% $\cdot$ T_{clk}; this result is a direct consequence of the smoother shape of the dynamic path distribution induced by AED-C, as explained in Section 4.3.1. In fact, even with the maximum value of the DW, the Vdd scales more aggressively than Razor (in average

0.96 V against 1.02 V) leading to errors miss-detection, thus output quality degradation.

The throughput of both Razor and AED-C shows a strict relation with ER_{th} as demonstrated in [126]. Since ER_{th} is the same (2%) the worst-case OPC is 0.98 (2% throughput degradation).

These results confirm that moving from Razor to AED-C enables an efficient Energy-Accuracy tradeoff. One may argue that an approximate version of Razor could have been obtained using the error-threshold ER_{th} as a control knob instead of tuning the detection window width DW. It is, but energy savings would have been much lower due to large performance penalties. The use of a larger ER_{th} implies the raise of timing errors to correct. More timing errors means that the number of cycles wasted for correction increases, hence the OPC decrease quickly with ER_{th}. To give a proof, the plots in Figure 4.30 show Vdd_{avg} and OPC as function of the error threshold ER_{th} for both FIR and IIR; ER_{th} is in the range [2%, 50%]. The workload is the one described above in this section. While ER_{th} keeps larger, the Vdd_{avg} reduces as expected: from 1.03 V to 0.70 V for FIR; from 1.02 V to 0.87 V for IIR. However, the performance loss is substantial as the OPC falls down: from 0.98 to 0.66 (almost two clock cycles for each operation) for FIR; from 0.98 to 0.67 for IIR. With AED-C the OPC overhead is 2% at worst.



(a) FIR Filter

(b) IIR Filter

Figure 4.30: Tuning ER_{th} for approximate Razor.

4.3.5 AED-C for Dual-Mode AVOS: a Pros&Cons Analysis

AED-C for Dual-Mode AVOS

The proposed AVOS leverages the AED-C to enable two operating modes: *Slow* and *Aggressive*. The former forces the Vdd scaling to be slow as it ensures full detection of all the timing violations within the first half of T_{clk} (TDW=50%·T_{clk}); this mode mimics a standard Razor AVOS scheme. The latter accelerates the Vdd scaling using a smaller TDW and hence a weaker error detection; under this mode some timing faults are not covered by the detection window and may result undetected with a negative impact on

the quality of the outputs. The decision to switching from/to *Slow* to/from *Aggressive* is demanded to the power-management unit (out of the scope of this work).

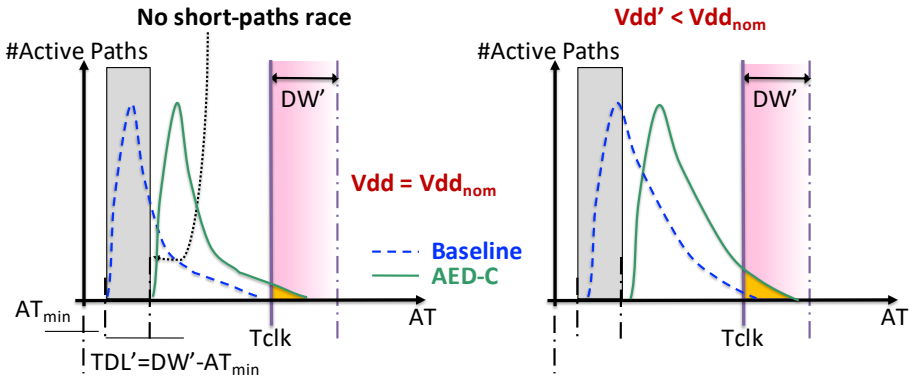


Figure 4.31: AED-C in *Slow* AVOS mode.

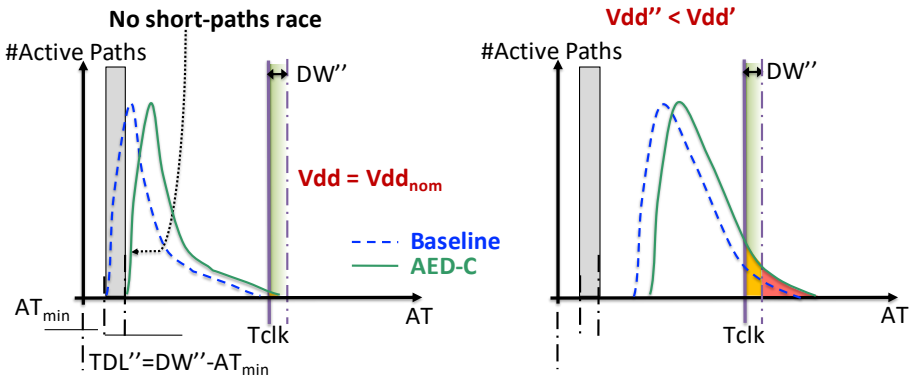


Figure 4.32: AED-C in *Aggressive* AVOS mode.

Figure 4.31 and 4.32 show an abstract dynamic path distribution (profile of the activated paths vs. their arrival time *AT*), to pictorially explain the working principle of the two modes. For both the figures dashed lines refer to a generic circuit after synthesis, i.e. w/o AED-C, while the solid lines refer to the same circuit after the TDL insertion, i.e. w/ AED-C.

Slow mode: Figure 4.31 left shows the dynamic path-distribution at nominal Vdd. Due to the presence of the TDL, there exist sensitized paths beyond T_{clk} (yellow area) which induce timing errors. However, for the rule of thumb introduces in the previous section, these errors are small in number. The Vdd can be therefore lowered until also the shorter and more active paths violate T_{clk}; at this point, Vdd' in Figure 4.31 (right), the Vdd scaling stops, and circuit operates at minimum energy. Since TDW is taken pretty large, any error beyond T_{clk} is properly detected and corrected (in Figure 4.31 right TDW covers all the paths); this ensures a slower, yet "safer" AVOS.

Aggressive mode: the width of TDW is reduced and so that of the TDL according to

Equation 4.3). As for the *Slow* mode, the Vdd is let scaling down from its nominal value, Figure 4.32 (left), till the minimum energy point, Vdd'' in Figure 4.32 (right). Since the error detection capability is lower (yellow area considerably smaller than in the *Slow* mode), a much lower number of errors get sampled; this allows the circuit settling to a lower voltage ($Vdd'' < Vdd'$). A direct consequence of such more aggressive Vdd scaling is that the circuit may suffer from potential *undetected* errors. Indeed, specific sequences of input patterns might push the supply voltage so down (Vdd'') that some paths may run in *off-side*, i.e. beyond TDW (red area in Figure 4.32 left). These errors cannot be detected, not even corrected; that is why the technique is suited for error-resilient applications.

Undetected errors represent a source of silent QoR degradation. However, this undesired condition is statistically infrequent: *overall, the Vdd scaling is kept to safe values by the most active paths which serve as a flywheel that guarantees convergence.* Indeed, the proposed AED-C exploits the characteristics of circuits that operate real-life workloads where data maintain a certain spatial/temporal correlation. It is unlikely that short-paths (those behind the detection window) and the off-side paths (those beyond the detection window, which show a very low switching probability) get active in sequence w/o any activation of the covered paths (which are those with the highest switching probability). As a matter of fact, there is a much higher probability that detectable errors come out and bring the system to a safe region (higher Vdd, larger DW) where undetected errors represent rare events. In other words, the convergence of the systems is ensured by those paths with higher activation probability; such paths always switch within the detection window thereby moving the Vdd to a proper safe value. Evidence of this principle are given in the next experimental section.

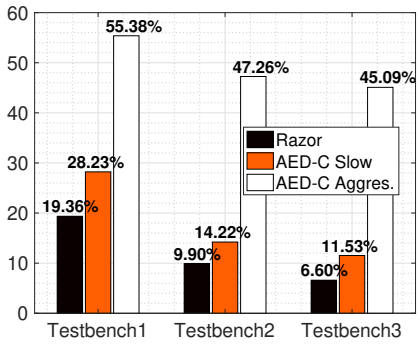
Experimental Results

The analysis reported in this section elaborates on two specific benchmarks, the *FIR* and *IIR* filters, which represent two corners in the efficiency spectrum of AED-C, as shown in the previous section. We used as testbenches the same three baseband audio signals but taken independently with $5 \cdot 10^6$ samples each. The context scenarios are the same: The $Testbench$ 1 is a noiseless voice recording; the switching activity of the LSBs is very low for a long portion of the stream. the $Testbench$ 2 is taken from an office conversation recording; samples present low noise and inputs have homogeneous switching activity. $Testbench$ 3 is an outdoor conversation; the recording is noisy and the switching activity of the inputs irregular.

This analysis is conducted on the Razor implementation (DW = 50%·T_{clk}) and the AED-C implementation (DW = 50%·T_{clk} for *Slow* mode, 25%·T_{clk} for while for the *Aggressive* mode, as per the study reported in the previous sub-section). The table in Figure 4.33a gives an overview of the quality metrics (section 4.4.1) collected for the *FIR Filter*. When operating in *Slow* mode, AED-C shows similar Vdd scaling capability of Razor (see Vdd_{min} and Vdd_{avg} columns) ensuring error-free outputs ($UE = 0$ for all

ED-C	Vdd_{min} [V]	Vdd_{avg} [V]	EPO *	OPC	UE [ppm]	NRMSE [%]
Testbench 1						
Razor (50%)	0.84 V	0.98 V	0.806	0.979	0	0
AED-C (Slow)	0.84 V	0.96 V	0.718	0.980	0	0
AED-C (Aggres.)	0.66 V	0.78 V	0.446	0.981	39	0.17
Testbench 2						
Razor	0.98 V	1.02 V	0.901	0.979	0	0
AED-C (Slow)	0.96 V	1.00 V	0.858	0.981	0	0
AED-C (Aggres.)	0.78 V	0.84 V	0.527	0.980	1	0.04
Testbench 3						
Razor	0.98 V	1.04 V	0.934	0.972	0	0
AED-C (Slow)	0.96 V	1.02 V	0.885	0.981	0	0
AED-C (Aggres.)	0.80 V	0.85 V	0.549	0.980	0.4	0.01

(a) AED-C vs Razor



(b) EPO Savings w.r.t. Baseline

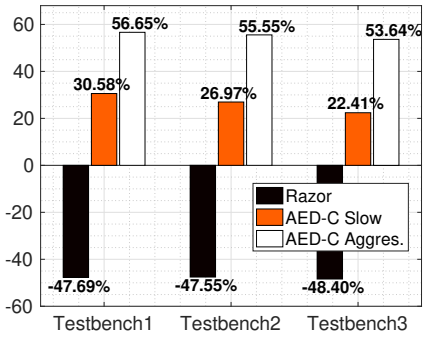
Figure 4.33: FIR Filter QoR/Savings summary.

the testbenches). This proves AED-C (*Slow*) does not affect the detection-correction responsiveness of the system. However, while Razor suffers power penalties due to static short-path padding, AED-C (*Slow*) achieves a much lower EPO. As shown in Figure 4.33b, the energy savings (w.r.t. Baseline) are above those of achieved with Razor: 28.2% vs. 19.4% for the best case ($Testbench$ 1). Even more impressive results are obtained when considering the AED-C *Aggressive* mode. While for Razor the Vdd_{avg} spans the range [0.98 V - 1.04 V], for the AED-C implementation Vdd moves down in the range [0.78 V - 0.85 V]. Figure 4.33b shows energy savings with AED-C increase from 28.2% (*Slow*) up to 55.4% for best case ($Testbench$ 1), much larger than with Razor. Concerning QoR degradation, the analysis reveals UE is in the range [0.4 - 39] ppm, with a mere $NRMSE$=0.17% (worst case over all the testbenches). Obviously different testbenches bring to different UE rate and different output degradations. The results give emphasis to the statistical nature of the proposed method and demonstrate the convergence of the systems under different input data distribution: undetected errors are rare events and $NRMSE$ gets very small. The performance (OPC) degradation due to errors correction does not change much between Razor and AED-C (in both the operating modes); OPC is mainly affected by the Vdd-scaling policy, which is the same for both the techniques.

On the opposite corner, the results of the *IIR filter* (Figure 4.34) highlight how the efficiency of the proposed AED-C is may sensible vary depending on the structure of the circuit. The table of Figure 4.34a shows the short-path padding procedure is quite inefficient: Razor EPO is 1.48× larger than baseline at best ($Testbench2$). That's the effect of buffer insertion during short-path padding, which introduces huge power overhead (area penalty is 112% w.r.t. baseline circuits as reported in Sec.4.3.4). By contrast, AED-C shows remarkable EPO savings: 30.6% and 56.7% for *Slow* and *Aggressive* respectively in the best case ($Testbench1$). Nonetheless, $NRMSE$ suffers significant drops: 9.87% at worst case ($Testbench3$). As per our analysis, that's due to the internal topology of

ED-C	Vdd_{min} [V]	Vdd_{avg} [V]	EPO *	OPC	UE [ppm]	NRMSE [%]
Testbench 1						
Razor	1.00 V	1.02 V	1.477	0.974	0	0
AED-C (Slow)	0.92 V	0.95 V	0.694	0.980	0	0
AED-C (Aggres.)	0.74 V	0.77 V	0.434	0.979	88.93k	0.39
Testbench 2						
Razor	0.98 V	1.02 V	1.476	0.980	0	0
AED-C (Slow)	0.88 V	0.97 V	0.730	0.980	0	0
AED-C (Aggres.)	0.72 V	0.78 V	0.445	0.979	138,65k	9.87
Testbench 3						
Razor	1.00 V	1.02 V	1.484	0.977	0	0
AED-C (Slow)	0.92 V	0.99 V	0.776	0.979	0	0
AED-C (Aggres.)	0.74 V	0.80 V	0.464	0.980	121.87k	6.68

(a) AED-C vs Razor



(b) EPO Savings w.r.t. Baseline

Figure 4.34: IIR Filter QoR/Savings summary.

the circuit made up of feedback path: an undetected error may propagate back to the internal paths and persist until a sequence of input patterns mask it. That's why the number of uncovered logic errors gets larger ($138.65 \cdot 10^3$ ppm for $Testbench3$).

On the importance of Dynamic short-path padding

One might argue that a comparison between AED-C in *Aggressive* mode (DW=25%) and Razor (DW=50%) is unfair. However, it must be considered that Razor is implemented at design time, using special FFs with a fixed DW; the short-path padding method that makes Razor-FFs working properly is a static method as well. Therefore, the use of Razor with a smaller DW (e.g., DW=25%) is unpractical as it would imply a circuit that rarely works at maximum QoR. Moreover, even assuming a DW fixed at 25% could be acceptable, the reshaping of the path distribution posed by short-path padding has very negative effects on AVOS capability. We report a cross comparison upon all the benchmarks: *Aggressive* AED-C vs. Razor with fixed DW= $25\% \cdot T_{clk}$. The input patters used for the digital filters belong to $Testbench1$ (the one which has shown maximum EPO for Razor).

The results, collected in Table 4.7, clearly show a Razor with smaller DW improves the Vdd scaling getting closer to AED-C results. However, our AED-C strategy achieving larger EPO savings at the cost of a very slight increase of the $NRMSE$. Do not forget that $NRMSE$ can be recovered in AED-C by adjusting the DW; the same cannot using Razor and its static DW.

ED-C	Area Overhead [%]	Vdd_{avg} [V]	EPO saving [%]*	NRMSE [%]
MAC				
Razor (25%)	30.5	0.79	53.1	0.21
AED-C (Aggres.)	16.9	0.71	64.9	0.60
FIR Filter				
Razor (25%)	18.8	0.82	44.0	0.10
AED-C (Aggres.)	2.1	0.78	55.4	0.17
IIR Filter				
Razor (25%)	81.0	0.82	38.5	0.23
AED-C (Aggres.)	44.8	0.77	56.6	0.39

Table 4.7: *Aggressive* AED-C vs. Razor with DW= 25%·T_{clk}.

4.3.6 Image Processing Case Study: DCT in JPEG Compression

Adaptive energy-accuracy scaling strategies work well on those applications that show a certain degree of tolerance to errors. More specifically, on those applications where output errors do not affect, or weakly affect, the quality of results perceived by the end-users (usually humans). In this section, the proposed AED-C for Dual-mode AVOS is tested on a realistic error-resilient application for image processing: a *Forward Discrete Cosine Transform* (FDCT). This block is typically used for applications like *lossy compression* of audio signals (e.g. MP3) and images (e.g. JPEG). It consists of a 8×8 fully pipelined parallel DCT synthesized at f_{clk} = 600 MHz. The AED-C timing monitors are placed according to the design flow described in 4.3.3. The Vdd-scaling policy is the one described in Section 4.3.2: monitoring period N=1000 clock cycles; ER_{th} fixed to 2% of the error rate.

Area Overhead

Table 5.2 collects the statistics of the AED-C implementation. Column **#FFs** reports the total number of sequential cells, while column **#R-FFs** the percentage of timing-critical FFs replaced with timing monitors of Figure 4.21. The last two columns show the

ED-C Technique	Area [μm^2]	#FFs	#R-FFs	Total Area Overhead	Dyn. SPP Area Overhead
AED-C	64873	4885	50.4%	21.8%	4.9%

Table 4.8: AED-C based FDCT Area Overhead.

overall area overhead and the area overhead due to Dynamic short-path padding implemented with the TDLs; both are normalized w.r.t. the baseline circuit (i.e. design w/o any AED-C mechanism). The area overhead (21.8%) is mostly due to the high number of AED-C monitors (16.9%), while the contribution due to the TDLs insertion is marginal (4.9%).

Simulation and QoR

In order to evaluate the energy/accuracy achievements brought by AED-C, the FDCT is embedded in a standard JPEG compression/decompression architecture implemented in *Matlab-Simulink*. The image resulting from the JPEG process is then compared with a reference image, i.e., the image obtained using the baseline FDCT (w/o any ED-C mechanism). The QoR is expressed using the *Peak Signal-to-Noise Ratio* (PNSR):

$$PSNR = 10 \cdot \log_{10}\left(\frac{peak_I^2}{ImMSE}\right) \tag{4.5}$$

The $peak_I$ is the maximum possible pixel value of the image (i.e., 255 in our case). The $ImMSE$ is the *Image Mean-Squared-Error*:

$$ImMSE = \frac{\sum_{i=0}^{n}(y[i] - y_o[i])^2}{n} \tag{4.6}$$

where y is the pixel of the image obtained with the FDCT implemented using the AED-C, while y_o refers to the same pixel returned by the baseline FDCT; n is the total number of pixels. Subjective analysis reveal a $PSNR$ value of at least 35 dB is recognized by users as of good quality.

The Testbench consists of a set of 24 grayscale images HD-ready resolution (1024x768 pixels) taken from an open-source repository (*pexels.com*).

Fig 4.35 reports the histograms of Vdd_{avg} and EPO for all the 24 images processed by the FDCT. As expected, AED-C shows limited Vdd-scaling (and energy savings) when it operates on *Slow* mode. Table 4.9 collects average results collected from simulations; the voltage gap between *Slow* and *Aggressive* mode is 200 mV (220 mV) for Vdd_{min} (Vdd_{avg}). This leads different average EPO savings w.r.t. Baseline: 12.5% (*Slow*), 51.9% (*Aggressive*). Concerning performance loss, the Table 4.9 shows OPC degradation is below 1% for both the operating modes.

AED-C	Vdd_{min} [V]	Vdd_{avg} [V]	EPO *	OPC
Slow mode	0.95 V	1.04 V	0.875	0.979
Aggressive mode	0.75 V	0.82 V	0.481	0.981

Table 4.9: AVOS efficiency over the set of pictures.
**Note: normalized w.r.t. Baseline*

In terms of QoR, the *Slow* mode ensures zero undetected errors, and hence uncorrupted JPEG images, while *Aggressive* mode affects output quality only marginally. Figure 4.36 reports the $PSNR$ distribution for all the 24 images composing the test-set: min-max range is [42.06 - 52.55],dB (best case), average 48.45 dB. The worst and the best

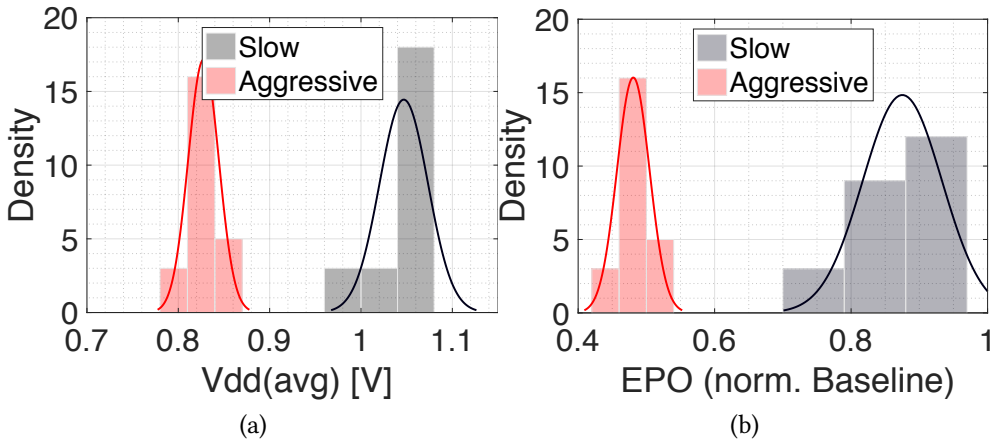


Figure 4.35: Vdd_{avg} and EPO distrubutions

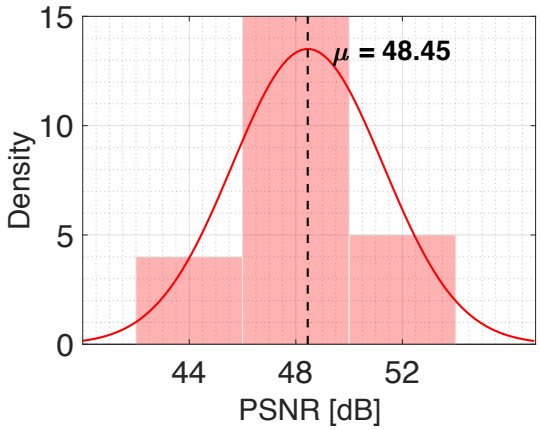
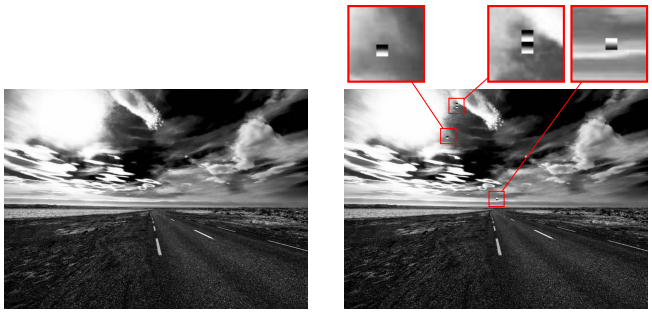


Figure 4.36: PSNR distribution over the testbench set of images.

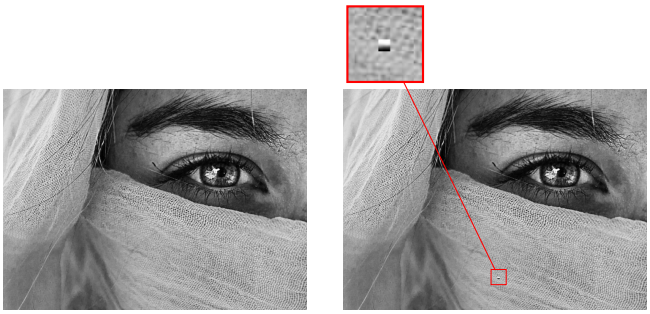
cases are reported in Figure 4.37 and Figure 4.38 respectively; both of them show tiny imperfections, as only few 8×8 pixel blocks are corrupted. The EPO-savings/$PSNR$ ratio is 57.5% / 42.06 dB for Figure 4.37b, 49.2% / 52.55 dB for Figure 4.38b. The number of UE is 27 ppm on average, 4 ppm for the best case and 328 ppm for the worst case.

It is worth emphasizing that the error-resilient nature of the JPEG application plays an important role. As a matter of fact, several errors on the high-frequency components of the DCT are covered by JPEG compression mechanism (e.g., zonal masking + quantization). That's a typical case for which the *Aggressive* mode is particularly suited.



(a) Baseline (b) AED-C Aggressive

Figure 4.37: Worst Case JPEG image.



(a) Baseline (b) AED-C Aggressive

Figure 4.38: Best Case JPEG image

Battery Lifetime Gain Analysis

As a final remark, we also describe a practical use-case: *A user is making an extensive use of a multimedia app (e.g., video streaming, gaming, etc.) on his portable device. Since the battery is running low, the power-management unit might decide to reduce the video quality to gain battery lifetime.* This context perfectly matches the case where a switch from *Slow* to *Aggressive* mode might help to meet the requirements imposed by the context.

The plot reported in Figure 4.39 shows the battery lifetime gain when the FDCT is made switching from *Slow* to *Aggressive* mode at a given percentage of battery. The bars are normalized w.r.t. *Slow* mode. When the switch threshold is set to 30%, the average lifetime extension is 24.6%, while for 70%, the lifetime extension will increase up to 73.7%. If the user wants to preserve battery since the very beginning (threshold 100%) the overall lifetime increases up to 81.9% (lifetime almost doubled).

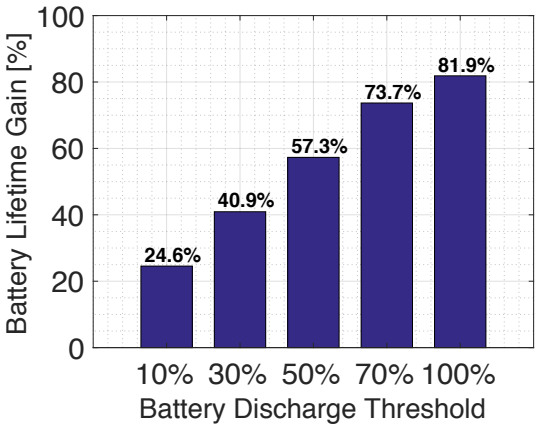


Figure 4.39: Dual-mode AED-C for battery lifetime average gain.

4.4 Adaptive EAS strategies comparison: ANT Versus AED-C

The broad objective of this section is to provide a fair comparison between AED-C and ANT (please, refer to section 3.4.2). A parametric characterization conducted over a set of realistic applications quantifies several figures-of-merit, like energy savings, performance and area overhead. The benchmarks consist of two digital filters, a FIR and a IIR, both synthesized and mapped onto a commercial FD-SOI CMOS technology at 28nm. The FIR is a pipelined 16^{th}-order low-pass filter in the direct form (12-bit in, 24-bit out) synthesized to f_{clk} = 650 MHz. The IIR is a pipelined 8^{th}-order low-pass filter in direct form I (16-bit in, 32-bit out) synthesized to f_{clk} = 650 MHz. The results collected for a sequence of three different classes of baseband audio signals empirically disclose the efficiency of ANT and AED-C, also providing an assessment of the resulting energy-quality tradeoff [128]. The benchmarks have been simulated using a sequence of three different baseband audio signals ($5 \cdot 10^6$ samples in total) as in Section 4.3.4.

4.4.1 Parametric Analysis

Figures of Merit

The comparison between AED-C and ANT includes the following metrics:

- Vdd_{avg}: the average Vdd obtained during testbench Voltage over-scaling simulation for AED-C based timing speculation. For RPR-ANT, the average voltage corresponds to the Vdd employed during the testbench simulation.
- *Energy per Operation* (EPO): the ratio between energy consumed and number of operations completed.
- *Operation per Clock Cycle* (OPC): the ratio between the number of executed operations and total number of clock cycles, considering that in AED-C techniques error corrections through logic masking need a cycle of clock gating. For RPR-ANT the OPC will be always 1, since no performance loss are conceived.
- *Normalized Root Mean Squared Error* (NRMSE):

$$NRMSE = \sqrt{\frac{\sum_{i=0}^{n}(y[i] - y_o[i])^2}{n}} \cdot \frac{1}{\mid max(y_o) - min(y_o) \mid} \tag{4.7}$$

  with y the value sampled at the output of the circuit, y_o the right output value, n is the total number of operations; the absolute value of the the max. and the min. value of y_o difference defines the output dynamics range. $NRMSE$ quantifies the quality of results.

- *Maximum Absolute Error* (MAE): expressed in $\log_2$ form,

$$MAE = \log_2 \left\lfloor \max_{input\ patterns} \mid y[i] - y_o[i] \mid \right\rfloor - 1 \tag{4.8}$$

  with y the value sampled at the output of the circuit, y_o the right output value. This metric representation collapses the maximum error on a single bit of the output.

Qualitative analysis

As already explained in Section 3.4.4, under the accuracy point of view, ANT and AED-C can be associated with two distinct class of approximate techniques: ANT belongs to the class of *Fail small* applications. The errors introduced by the voltage scaling remain "small" in magnitude (as it is bounded by the precision of the replica circuit) but quite frequent. AED-C belongs the class of *Fail Rare*. The error magnitude is usually pretty large (the long timing paths of arithmetic circuits are commonly on the MSB of the output) but very infrequent (long timing paths are activated rarely). This separation reflects the difference between voltage scaling using *output compliance* and voltage scaling using *timing compliance*. The quantitative analysis of the next section confirm this trend through a more concrete comparison.

Quantitative analysis

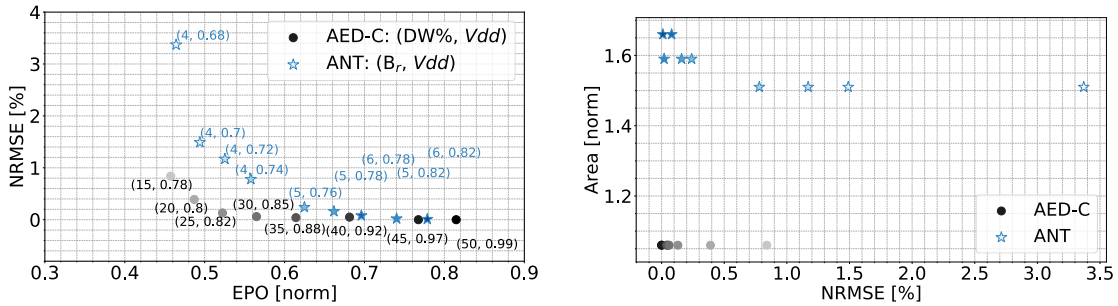
To perform a formal comparison between ANT and AED-C, figures of merit such as quality of results, energy savings, performance and area overhead were assessed. Since there are too many possible design variables and settings to show, this section provides a more compact, yet complete Pareto analysis.

Let us first analyze the FIR filter. Figure 4.40a shows the Pareto points in the quality vs. energy space ($NRMSE$ vs. EPO). The AED-C points (black dot) are labeled with the caption *(DW%, Vdd_{avg})* and the ANT points (blue stars) are labeled with *(B_r, Vdd)*, with B_r as the number of bits of the replica circuit and Vdd is the operating voltage. It is worth noting that, in the ANT case, the valid Pareto points are only those whose operating voltage ensures no timing violations in the replica circuit.

As a general trend, AED-C showed a better energy–quality tradeoff, except for the rightmost points at DW=50% of T_{clk}, which was dominated by the ANT point (6, 0.82). The results can be simply explained considering that the ANT architecture required a more approximated replica circuit, i.e., lower B_r, to achieve the same energy savings of AED-C; however, a too approximated replica induced a quick increase of error.

To better appreciate this trend, Table 4.10 gives a more detailed view. In the first row (*Min. EPO point*), it shows the comparison between the points of ANT and AED-C with the highest energy-efficiency: AED-$C_{(15,\ 0.78)}$ and $ANT_{(4,\ 0.68)}$. For almost the same energy, AED-C gave results of a much higher quality ($NRMSE$ = 0.84% for AED-C vs. 3.37% for ANT). Evidence of the AED-C superiority is also given by looking at the second row (*NRMSE-EPO Knee point*), which collects the metrics for ANT and AED-C across the knee of their $NRMSE - EPO$ Pareto curves: AED-$C_{(25,\ 0.82)}$ and $ANT_{(5,\ 0.78)}$. For almost the same quality of results, AED-C outperformed ANT in terms of energy savings (EPO = 0.52 for AED-C vs. 0.66 for ANT). Even though the ANT implementation reached a lower Vdd, its energy savings was limited by the architectural overhead. This aspect emerged clearly from the Area–Quality Pareto analysis of Figure 4.40b. For the sake of clarity, it should be noticed that at best accuracy Pareto points, as reported in Table 4.10 third row (*Min. NRMSE point*), ANT presented slightly lower EPO than AED-C, i.e. 0.78 vs. 0.81. When accuracy is the priority, the DW should be taken larger. For such conservative case, ANT was more efficient than AED-C, in which the activation of the longest paths limited the voltage scaling (hence, the energy savings).

To ensure a specific output quality, the replica circuits need more arithmetic precision and they take huge silicon area. For minimum output degradation, the area overhead reached with ANT was more than 60%, too much for real-life circuits. It is worth noting that AED-C showed a constant area overhead as the different configurations were obtained only by tuning the TDW/TDL width, with no micro-architectural modifications. Not just area, also throughput (OPC) needs special care. Table 4.10 shows that ANT did not suffer any performance penalty, while AED-C was 2% slower due to error detection-corrections. It is worth highlighting how the distribution of the timing



(a) Quality vs. Energy

(b) Area overhead vs. Quality

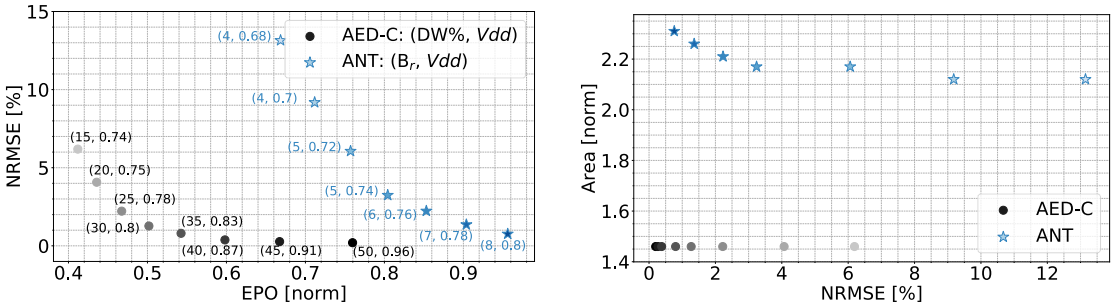
Figure 4.40: ANT vs. AED-C timing speculation: FIR filter.

Technique	Vdd [V]	EPO [norm]	OPC	N. Errors	NRMSE [%]	MAE [norm]	Area [norm]
			Min. EPO point				
AED-C (DW = 15%)	0.78	0.46	0.98	3226	0.84	21	1.06
ANT (B_r = 4)	0.68	0.46	1.00	1,382,023	3.37	17	1.51
			NRMSE-EPO Knee point				
AED-C (DW = 25%)	0.82	0.52	0.98	114	0.13	20	1.06
ANT (B_r = 5)	0.78	0.66	1.00	97,664	0.16	16	1.59
			Min. NRMSE point				
AED-C (DW = 50%)	0.99	0.81	0.98	16	0.01	17	1.06
ANT (B_r = 6)	0.82	0.78	1.00	11,746	0.01	16	1.66

Table 4.10: Quantitative comparison summary: FIR filter.

errors for both techniques followed the classification made at the beginning of this section. As reported in Table 4.10, AED-C was characterized by a lower number of errors (col. *N. Errors*, over 5×10^6 input patterns) of high magnitude (col. *MAE*); the opposite held for ANT, namely more errors of lower amplitude. Although the magnitude of the errors in AED-C is higher than ANT, the effect on the overall quality of results remain acceptable (col. *NRMSE*).

The comparative analysis performed on the IIR filter emphasized even more what the FIR analysis showed. As reported in Figure 4.41a and Table 4.11, AED-C guaranteed higher energy efficiency than ANT and all the ANT implementation were dominated by AED-C. A similar consideration done for FIR still held. Although ANT pushed the supply voltage to lower value, it still could not achieve the same energy of AED-C. Shrinking the ANT replica circuit to B_r = 4 (point $\text{ANT}_{(4,\ 0.68)}$), the EPO became close to that reached with $\text{AED-C}_{(45,\ 0.91)}$, yet with an unacceptable output degradation ($NRMSE$ 13.15% vs. 0.27%). Conversely, for the same $NRMSE$, the area overhead became too large for a realistic implementation (Figure 4.41b).



(a) Quality vs. Energy

(b) Area overhead vs. Quality

Figure 4.41: ANT vs. AED-C timing speculation: IIR filter.

Technique	Vdd [V]	EPO [norm]	OPC	N. Errors	NRMSE [%]	MAE [norm]	Area [norm]
			Min. EPO point				
AED-C (DW = 15%)	0.74	0.41	0.98	93387	6.19	30	1.46
ANT (B_r = 4)	0.68	0.67	1.00	4586948	13.15	26	2.12
			NRMSE-EPO Knee point				
AED-C (DW = 35%)	0.83	0.54	0.98	11759	0.80	30	1.46
ANT (B_r = 5)	0.72	0.76	1.00	2316127	6.01	25	2.17
			Min. NRMSE point				
AED-C (DW = 50%)	0.96	0.76	0.98	1548	0.20	27	1.46
ANT (B_r = 8)	0.80	0.96	1.00	99872	0.76	21	2.31

Table 4.11: Quantitative comparison summary: IIR filter.

4.4.2 Final Considerations

The proposed AED-C is based on the probabilistic assumption that an efficient voltage over-scaling can be achieved if long-paths are rarely activated. This is the same consideration under which both Razor and ANT are built. However, for our AED-C, there might be specific sequences of input patterns for which the Vdd is pushed so low that some paths run beyond the detection window, thus leading to potential miss-detected errors and output quality degradation. This is the main difference with respect to Razor and ANT (which are bounded in terms of quality degradation instead). This also reflects the limits of the AED-C technique: a too frequent activation of the longest paths may limit the voltage scaling and hence the energy savings. There is a trade-off between accuracy and savings. When accuracy is the priority, the Detection Window (DW) should be taken larger. For such conservative cases, ANT may outperform AED-C. This is shown in Figure 4.40a, where the FIR filter with DW = 45%(50%) $\cdot T_{clk}$ showed lower energy savings compared to ANT.

The key of AED-C is that it can be implemented with low design overhead. By contrast, ANT requires a replica circuit that introduces severe area (and hence energy) penalty.

Chapter 5

Static EAS via Inferential Arithmetic Circuits

The broad objective of this dissertation is to introduce advanced design solutions that improve the approach the EAS paradigm is implemented. This chapter deals with a new strategy for *Static* EAS. According to the formal definition given in Section 3.2, an EAS solution is static if the energy-accuracy tradeoff is fixed at design-time by functional speculation, i.e., a modification of the logic functionality through algorithmic or circuit simplifications which induce energy savings for a worst-case accuracy loss. In this thesis, the proposed solution involves the design of Inferential Logic Circuits whose logic functions can be described as inference rules [146]. Inspired by cognitive functions of the human brain, a machine learning-driven synthesis flows can map Boolean functions as Classification Trees that work like statistical inference engines. As explained in Section 3.2.2, circuits of this kind infer output values by evaluating the key features of the function learned during the training stage. The result is the design of combinational logic circuits that can mimic Boolean functions to a certain degree of accuracy. These inferential logic circuits run *quasi-exact* computations trading energy efficiency for accuracy in error-resilient applications.

The Inferential Logic principle has a positive impact on arithmetic applications [145] [147] where inferential circuits design can lead to architectures more prone to support aggressive adaptive power management. For this reason, this idea has been proposed for arithmetic circuits and, more specifically, for the design of an *inferential* 8-by-8 bit unsigned multiplier. Using as case-study an error-resilient image blending application, we quantify the most representative figures of merit, also giving comparison against a classical radix-4 multi-level implementation. Experimental results prove the *inferential* multiplier simplifies the circuit complexity reducing the area by 22%. Also, the inferential multiplier can exploit 2× latency reduction for power optimization guaranteeing 76% average accuracy.

This chapter firstly discusses the Multiplication by Inference concept under an abstract point of view. Then, a brief recall of Classification Trees training algorithm on

the Boolean domain is reported. The Machine Learning driven design flow for arithmetic circuits, the core of this chapter, is disclosed along with the experimental results on the Inferential Multiplier. The chapter is closed by a comparative analysis with a multiplier obtained through a classical approximate computing technique, i.e., function under-design, as explained in Section 3.2.1.

5.1 Inferential Multiplier: Theories, Methods and Tools

5.1.1 Multiplication By Inference: an abstract viewpoint

The idea of solving problems through inductive reasoning has paved the way to a remarkable revolution in the ICT ecosystem. That's the rise of machine learning (ML), a computing paradigm where machines replicate a few simple learning/reasoning mechanisms proper of the human brain [17].

In recent years ML attracted lot of interest in several industrial and scientific areas, including the microelectronics field. While most of the VLSI research is focused on mapping ML algorithms into efficient HW platforms, little effort has been spent on investigating how brain-inspired learning mechanisms can help to solve EDA problems. Although a few previous work account on the possibility of using ML for functional verification and testing [160, 53], the idea of a design flow through which logic circuits can be "accelerated" by means of ML tools is covered only marginally.

The human brain works like a statistical inference engine [24, 148] that generalizes problems by means of experience-based induction. Such a generalization process encompasses two main stages: (*i*) find out what are the most significant characteristics of the problem; and (*ii*) define logic relationships among those characteristics in order to infer the best solution, ideally the *true* solution. To better understand this concept, let us consider how our mind works when solving a simple math question: what is the answer to $X \cdot 100$, with $X \in \mathbb{N}$? We know from previous experience, i.e., through inductive reasoning, that the result of any natural number multiplied by 100 (the key characteristic of the problem) is simply X followed by a trailing "00" (the logical process). Therefore, the yield, i.e., the *true* answer in this case, is obtained by skipping the arithmetic operation. Indeed, such process can be regarded as an inference process rather than an arithmetic one. Generalizing this simple, yet meaningful example, it is possible to assume that the human brain accelerates, or skips, some simple operations by extrapolating significant pieces of information from the complete picture of the problem.

In this chapter, we discuss the possibility to mimic such a computing paradigm by leveraging ML techniques during the design stages of logic circuits. More specifically, we raise the question: *is it possible to design a multiplier that works like an inference engine?* To answer this question we thoroughly describe and analyze an 8-by-8 bit unsigned multiplier obtained through a ML-driven synthesis flow plugged into a commercial logic synthesis framework. Circuit description is derived by shifting the logic

function of the multiplier towards a binary classification problem upon which a statistical representation of the circuit is learned.

5.1.2 Logic Inference Through Classification Trees

Background on Classification Trees

Solving a classification problem means to find the best abstract representation of the key characteristics, better known as features, over a given set of labeled observations. As reported in Figure 5.1a, the observation data-set, or training-set, consists of n different samples $\vec{s}_k$, with $1 \leq k \leq n$, where each sample is a tensor described by p predictor variables $X = \{x_1, \ldots, x_p\}$ and one label corresponding to one of the m available classes $y_i \in Y = \{y_1, \ldots, y_m\}$.

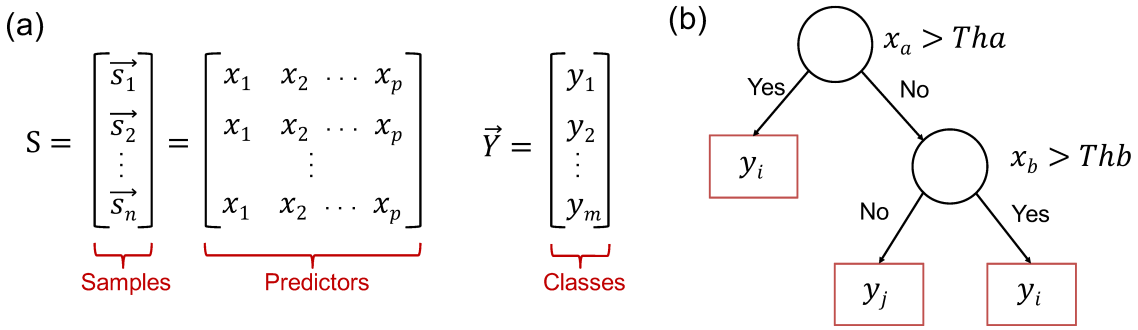


Figure 5.1: Classification problem: (a) analytical model, (b) abstract model representation using a Classification Tree.

A classification problem is typically represented using a compact abstract model built through statistical evaluation of the labeled observations; this stage is called *training*. The obtained model is used to classify new never-occurred samples $\hat{\vec{s}}_k$ described through the same predictors X; this stage is the *inference*. Good abstract models are those that achieve high quality-of-result (QoR), i.e., high classification accuracy, during the inference stage.

Although several options for building abstract models are available, Classification Trees (CTs) have proven to achieve high QoR with low level of complexity in many application cases [15]. As depicted in Figure 5.1b, a CT is a rooted and directed acyclic graph (DAG) defined as $\Gamma = (\Phi \cup D \cup \Theta \cup E)$. The root node $\phi \in \Phi$ with in-degree 0 represents the classification output. Terminal nodes $\theta \in \Theta$ with out-degree 0 represent the class $y_i \in Y = \{y_1, \ldots, y_m\}$ to which the root is associated. A decision node $d \in D$ implements a comparison between a predictor $x_j \in X$ and a threshold value T_h. During the traversing of the CT, each decision node activates one of the two outgoing edges $e_0, e_1 \in E$ in order to enable logic paths towards nodes at the lower levels. When a predictor x_j gets larger than its associated threshold T_h, the right branch e_1 is activated;

e_0 is activated otherwise. Both x_j and T_h are defined during the *training* phase by means of a recursive splitting of the observation data-set.

The building of the tree is driven by a statistical dispersion index, called *Gini* index, used to decide the best splitting predictor over the many available. More in detail, the *Gini* index is a parameter that measures how much a predictor is able to separate observations into different groups having the lowest level of impurity [47]. Since CTs are obtained through recursion, a stopping criteria must be defined. Indeed, if the algorithm is left free to evolve, the model eventually finds splits that are completely homogenous, yet with a trivial sample size. In order to prevent this behavior, also known as *overfitting*, the training algorithm stops when an *a-priori* minimum sample size is reached. As a result, only a subset of the available predictors are accounted in the final CT structure; such predictors are the most significant ones, i.e., those that bring the most valuable piece of information for the resolution of the classification problem.

Boolean Functions as Binary Classification

A classification problem where both predictors and labels belong to a Boolean domain is commonly referred to as a *binary* classification problem. To better understand how binary classification can be exploited in the context of logic synthesis, let us consider a generic Boolean function $\mathcal{F} : \mathbb{B}^n \rightarrow \mathbb{B}$, with support-set $S = \{x_1, \dots, x_n\}$.

The i-th permutation between primary inputs in S represents the product term $\mathcal{P}_i : \mathbb{B}^n \rightarrow \mathbb{B}$, where $\mathcal{P}_i \subseteq \mathcal{F}$; a collection of all product terms $(\mathcal{P}_1, \dots, \mathcal{P}_k)$, with $k = 2^n$, represents the truth table of $\mathcal{F}$. A truth table *per sé* is a binary data-set of observations upon which a CT structure representing important predictors (a subset of primary inputs) and their logic relationship can be derived [145].

Since the Gini index is applied on binary attributes, a key simplification takes place: threshold values selected during splits are the mean over the possible values assumed by attributes (i.e., 0 or 1), namely, T_h=0.5. Hence, as the visual equivalence depicted in Figure 5.2 suggests, the resulting decision nodes reduce to simple Boolean comparators, just like decision nodes in Binary Decision Diagrams.

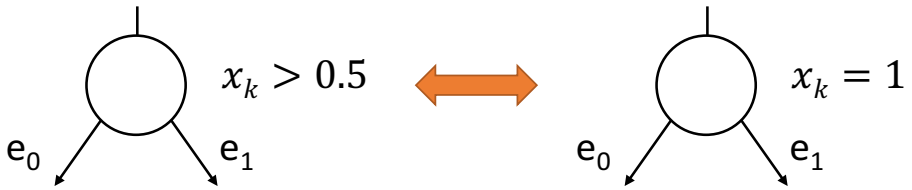


Figure 5.2: CT decision node: node threshold equivalence.

To better understand these concepts, let us resort to a practical example. Consider the Boolean function $\mathcal{F} : \mathbb{B}^3 \rightarrow \mathbb{B}$ defined by the truth table in Table 5.1. From an EDA perspective, if we consider the cube representation of a function $\mathcal{F}$, the classification rule that partitions the input space can be easily identified: a vertical plane centered

x_1	x_2	x_3	$\mathcal{F}$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Table 5.1: Truth table of function $\mathcal{F}$.

in $x_1 = 0.5$, as reported in Figure 5.3 (center). Therefore, the resulting Boolean function is $\widetilde{\mathcal{F}} = \overline{x1}$, the cube notation of which is depicted in Figure 5.3. We refer to this function as the quasi-exact representation of $\mathcal{F}$, since it covers six over eight minterms of the original function (highlighted bullets in Figure 5.3 (left) represent misclassified minterms).

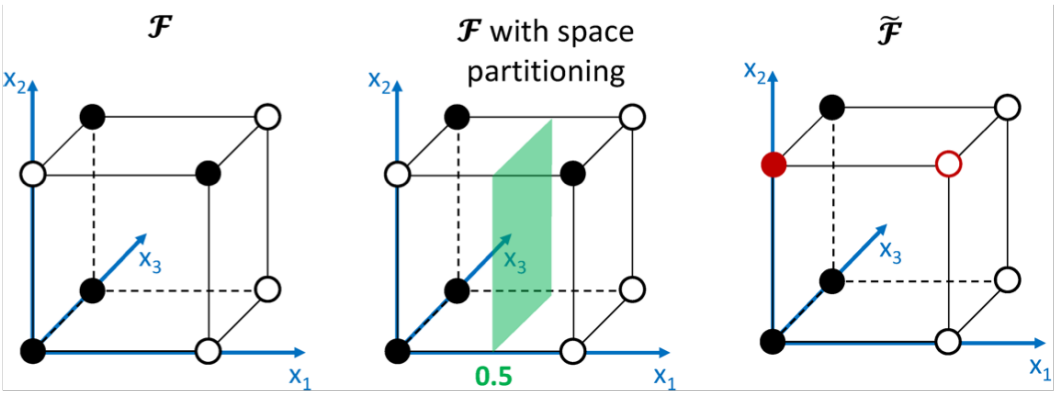


Figure 5.3: Cube notation. Original Boolean function $\mathcal{F}$ (left), input space partitioning due to classification tree (center), and resulting quasi-exact function $\widetilde{\mathcal{F}}$ (right) [147].

By learning the CT, it is therefore possible to derive a canonical representation of the function $\widetilde{\mathcal{F}}$, with $\widetilde{\mathcal{F}} \subseteq \mathcal{F}$, described over a reduced support-set $\widetilde{S} \subseteq \mathcal{S}$. Indeed, $\widetilde{\mathcal{F}}$ is typically less complex than $\mathcal{F}$. Intuitively, $\widetilde{\mathcal{F}}$ is not an exact representation of the original $\mathcal{F}$, it just covers a subset of all possible patterns instead. This opens to some circuit optimization for error-resilient applications, where accuracy is traded for speed and/or power consumption.

5.1.3 Machine Learning Driven Logic Synthesis Flow

The design automation of an *inferential* circuit based on CTs imposes some changes in the standard synthesis flow. As depicted in Figure 5.4, an additional stage for CT building is inserted just before multi-level logic synthesis&optimization. This new stage consists of two main tasks.

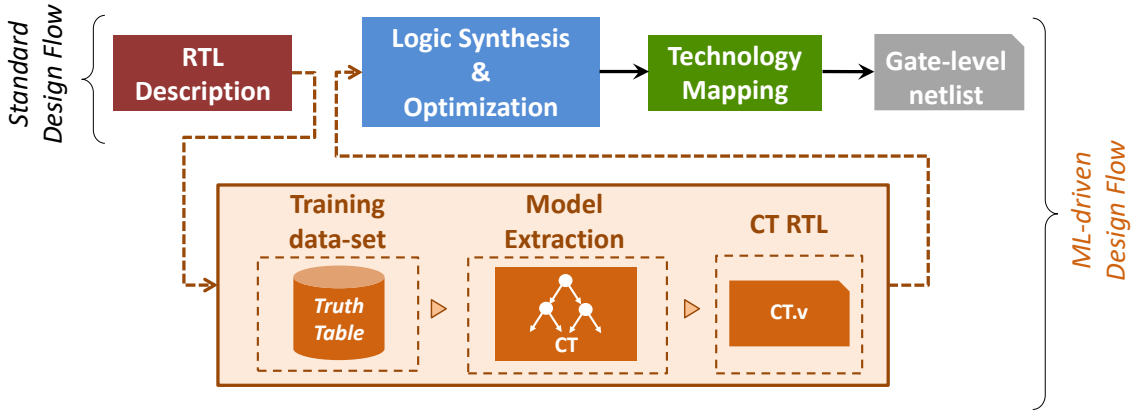


Figure 5.4: Proposed flow plugged into a standard synthesis flow.

1. Generation of the observations data-set: the RTL description is processed by means of a SAT solver that, for each permutation of the primary inputs, defines the corresponding output. The obtained table serves as data-set to build the CT.

2. CT-based model extraction: this is the core of the ML-driven synthesis flow; the function $\widetilde{\mathcal{F}} : \mathbb{B}^k \rightarrow \mathbb{B}$, with $k \leq n$, is learned through the process described in Section 5.1.2. The truth table is processed by a in-house *Matlab* script that leverages the built-in `fitctree` function. Multiple output functions $\mathcal{F} : \mathbb{B}^n \rightarrow \mathbb{B}^m$ are decomposed into multiple logic cones, each of them modeled by a dedicated CT. The resulting $\widetilde{\mathcal{F}} : \mathbb{B}^k \rightarrow \mathbb{B}^m$, where $k \leq n$, consists of an an ensemble of independent CTs. The obtained structure is then annotated in Verilog format and fed as input to the *logic synthesis & optimization* engine that run standard area, power, and delay minimization.

It is worth to emphasize that integrating the ML-driven stage into commercial tools is an easy task; we plugged the proposed ML-based flow into Synopsys Design Compiler through a dedicated wrapper consisting of Tcl and Matlab scripts.

5.1.4 Experimental Results

Benchmark and Experimental Setup

As test case we introduce the design of an 8-by-8 bit unsigned multiplier. This benchmark finds applications in many HW accelerators for image processing and classification in the context of visual reasoning. To better appreciate the characteristics of the CT-based circuit we provide a comparison between two possible implementations:

Inferential multiplier (I-MULT): the benchmark is optimized with the ML-driven synthesis flow described in Section 5.1.3 (Figure 5.4) plugged into Synopsys Design Compiler.

Radix-4 Booth multiplier (R4-MULT): obtained using a standard synthesis flow implemented into Synopsys Design Compiler and a standard-cell library, with low power and area features enabled.

For both the implementations, the logic mapping is done with an industrial FDSOI CMOS technology at 28nm. The frequency constraint is set to F_{clk} = 500 MHz.

Inferential vs. Arithmetic Multiplier

Table 5.2 reports the post-synthesis results for the two implementations. The I-MULT shows a simplified logic network (1.33× fewer devices) thus ensuring 22% less area w.r.t. the R4-MULT. Concerning performance, the I-MULT shows a wort-case timing slack that is 56% of the clock period (T_{clk}). If compared against the slack obtained with the R4-MULT, a mere 2.26% of T_{clk}, we can clearly state that the *inferential* arithmetic core is by far more efficient.

One may argue that the frequency constraint used for the synthesis is unfair as it favors I-MULT. However, when the maximum frequency of the I-MULT is used as a constraint, i.e., F_{I-MULT} = 6.67 GHz, the synthesis process of the R4-MULT does not converge. The largest frequency for which R4-MULT reaches timing closure is $0.45\times F_{I-MULT}$; under such constraint, the R4-MULT area is 52% larger than that of the I-MULT. Here's why the reported analysis is for F_{clk} = 500 MHz, a common target for tightly coupled parallel architectures which are typically memory bounded, e.g., accelerators for image classification and visual reasoning [25].

Multiplier	Gates	Area [μm^2]	Slack [%T_{clk}]
I-MULT	401	298.17	56.05
R4-MULT	535	383.85	2.26

Table 5.2: Synthesis results, with columns **Gates**, **Area**, and **Slack** representing the total number of logic gates, the total area, and the worst slack respectively.

Figure 5.5 shows the accuracy γ for each primary output of the I-MULT. Accuracy is defined as the ratio $\frac{E_c}{T_p}$, where E_c is the number of input patterns that are successfully classified, and T_p is the cardinality of the input permutation set, i.e., the number of rows in the truth table. Accuracy is measured through exhaustive functional simulation. The two MSBs ($m15, \ldots, m14$) and the five LSBs ($m4, \ldots, m0$) show the highest accuracy, $\gamma > 95\%$ and $\gamma = 100\%$ respectively, while lower accuracy is recorded for intermediate outputs ($m10, \ldots, m5$), for which $54.1\% < \gamma < 61.5\%$. This trend is due to the distribution of the binary classes $\{y_0, y_1\} \in Y$ in the training population. More specifically, the training set associated to MSBs and LSBs show a distribution highly skewed towards

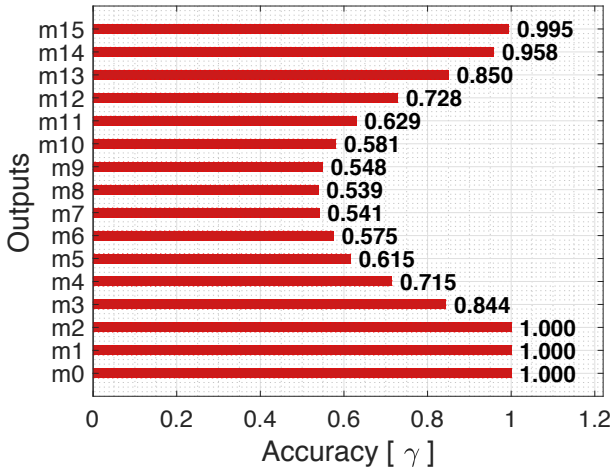


Figure 5.5: Output Accuracy of the I-MULT.

0 or 1. The learning algorithm is thereby prone to *overfit* the observation data-set [55]. While overfitting is an undesired effect in typical classification problems, it may represent a favorable condition to generalize Boolean logic functions. The overall accuracy averaged over all the outputs is 76%, a reasonable value for error-resilient applications as shown in the next subsection.

Image processing

We tested the efficiency of the proposed I-MULT on *Image Blending*, an image processing task in which two images, the *source* and the *mask*, are mixed for artwork. The computational kernel consists of a pixel-wise matrix multiplication. In our experiments, the testbench consists of 7 *source* and 8 *mask* HD-ready pictures with a resolution of 1024x768 pixels; the output set is then composed of 56 blended images.

The adopted evaluation metrics are described as follows.

Normalized Root Mean Squared Error (NRMSE): difference between predicted values, i.e., I-MULT results, and the expected results, i.e., those obtained with the R4-MULT at nominal conditions. The analytical equation is given by (5.1), where y is the value sampled at the output of the circuit, y_o is the right output value, n represents the total number of multiplications, and y_{max} (y_{min}) the max (min) value of y_o.

$$NRMSE = \sqrt{\frac{\sum_{i=0}^{n}(y[i] - y_o[i])^2}{n}} \cdot \frac{1}{y_{max} - y_{min}} \tag{5.1}$$

Peak Signal-to-Noise Ratio (PSNR): quality of a picture w.r.t. the expected one, i.e., those obtained with the R4-MULT at nominal conditions. PSNR is described in (5.2), where $peak_I$ is the maximum value assumed by each pixel, i.e., 255 when working with grayscale images, as in our case. Subjective analyses reveal PSNR in the range 20 dB to

35 dB are perceived as good results.

$$PSNR = 10 \cdot \log_{10}\left(\frac{peak_I^2}{MSE}\right) \tag{5.2}$$

Power consumption: the average power consumption of the circuit. As estimator we adopted the probabilistic model embedded into a sign-off tool (Synopsys PrimeTime) with signal statistics back-annotated from functional simulations by means of `saif` files.

When dealing with optimal energy/performance-vs-accuracy trade-offs, a common strategy is to trade power consumption for QoR using some power management technique, e.g., dynamic frequency scaling (DFS) and/or dynamic voltage scaling (DVS) [11]. Concerning DFS, Figure 5.6 gives a parametric analysis of power consumption (dotted line) and NRMSE (solid line) from F_{nom} = 500 MHz up to $2\times F_{nom}$. Notice that power consumption is normalized w.r.t. R4-MULT working at F_{nom}. Power consumption increases linearly with frequency. Within the whole frequency range, I-MULT results less power hungry than R4-MULT, even at $2\times F_{nom}$, where power savings is ~7%. Concerning QoR, the average NRMSE of I-MULT (solid line with square marker) is flat over the entire range, i.e., 5.45%, as no timing faults do occur. Even though that's a direct consequence of the available timing slack, it demonstrates that the I-MULT effectively accelerates the arithmetic computation with a small degradation of the result. The same cannot be said for R4-MULT (solid line with cross marker) which shows a substantial degradation (w.r.t. R4-MULT at nominal conditions) due to timing faults for operating frequencies larger than $1.18\times F_{nom}$; the worst case NRMSE = 20% at $1.82\times F_{nom}$. The break-even point is close to $\sim 1.33\times F_{nom}$, at which the I-MULT still shows significant power savings w.r.t. R4-MULT, i.e., 18%.

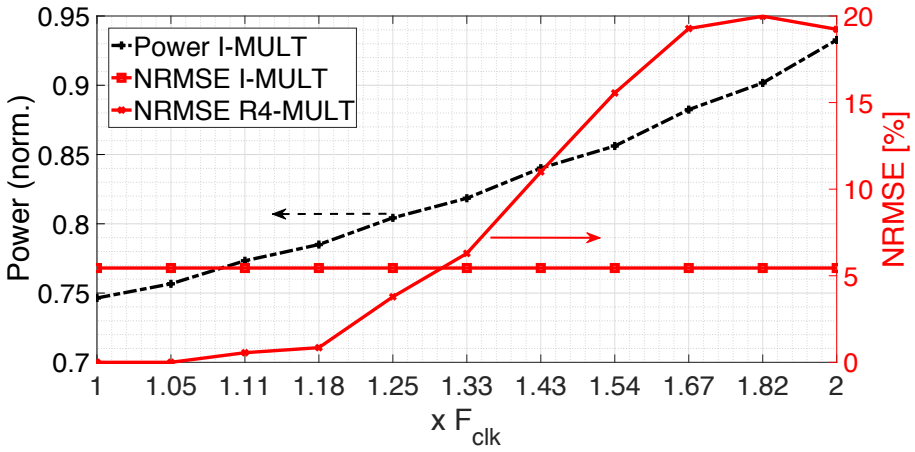


Figure 5.6: I-MULT: Power_{avg} and NRMSE_{avg} vs. F_{clk}

When playing with DVS, assuming as target frequency F_{nom} = 500 MHz, the timing

slack available with I-MULT can be consumed by reducing the supply voltage from the nominal value of 1.1 V to 0.64 V. With such a low voltage, the power savings w.r.t. R4-MULT (at F_{nom}) reach 80%, still guaranteeing low accuracy loss, NRMSE = 5.45%. It is worth to emphasize that for both DFS and DVS, the I-MULT implementation is timing-fault free. This prevents metastability at the rise-edge of the clock, an undesired effects of aggressive power management solutions typically adopted in approximate computing [20].

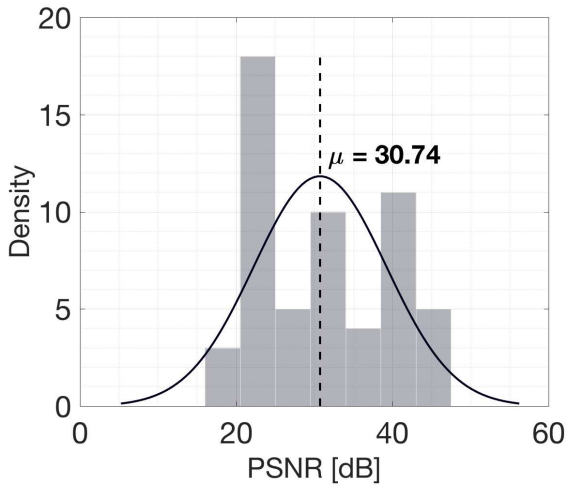


Figure 5.7: PSNR of the I-MULT on the 56 images

For the sake of completeness, Figure 5.7 reports the PSNR histogram for all the 56 output blended images processed by the I-MULT working at max speed, $2\times F_{nom}$ and 1.1V; the averaged PSNR is 30.74 dB, in the range of visually good results.

As a final remark, Figure 5.8 gives a visual representation of QoR degradation for two samples of the testbench. The pictures in Figure 5.8a and Figure 5.8b are the source and mask images respectively; the blended image is reported in Figure 5.8c. The pictures in Figure 5.8d and Figure 5.8e are the output of I-MULT and R4-MULT working at same frequency and voltage ($2\times F_{nom}$, 1.1V). While I-MULT achieves a good quality (PSNR = 31.24 dB), with the R4-MULT the output is clearly affected by errors (PSNR = 14.69 dB). This is a remarkable results if one considers that I-MULT is also more power and energy efficient.

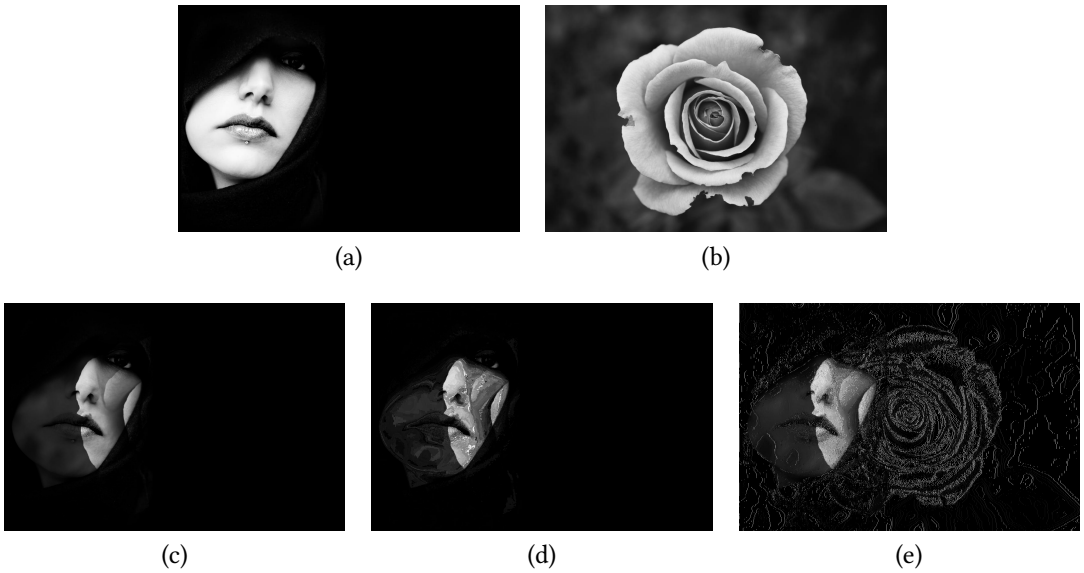


Figure 5.8: Image blending. Source (a) and mask (b) images, (c) exact image blending, (d) I-MULT result, (e) R4-MULT result.

5.2 Static EAS strategies comparison: Inferential vs. Approximate Multiplier

As shown in the previous section, the Inferential Multiplier (I- MULT) shows reduced circuit complexity, w.r.t. classical multiplier architecture, which leads to a latency compression exploited for power optimization. For the sake of completeness, a comparative analysis, in terms of area and accuracy, should be performed against Approximate Computing techniques, also. Specifically, we considered as a benchmark a representative Approximate Multiplier which we referred to as Kulkarni Multiplier (K-MULT). This multiplier is obtained through function under-design, i.e., by truth table simplification, as already discussed in Section 3.2.1.

In the following sections, we performed a parametric characterization in terms of area comparing I-MULT and K-MULT at their maximum frequency. Then, a study on the error distribution generated by both multipliers discloses how the nature of the system accuracy differs when *inferential* vs. *arithmetic* functional-speculation rules are applied.

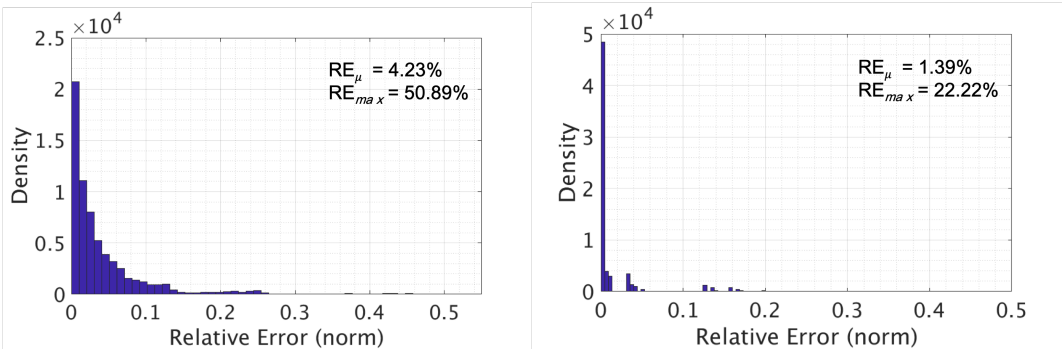
5.2.1 Hardware Characterization

In this analysis, we synthesized 8-by-8 bit unsigned I-MULT and K-MULT. For both the implementations, the logic mapping is done with an industrial FDSOI CMOS technology at 28nm. When the maximum frequency of the I-MULT is used as a constraint,

i.e., F_{I-MULT} = 6.67 GHz, the synthesis process of the K-MULT does not converge. The highest frequency for which K-MULT reaches timing closure is 0.55×F_{I-MULT}; under such constraint, the K-MULT area is 25% larger than that of the I-MULT. Thus, an inference engine is more area/latency efficient than both standard and simplified multipliers.

5.2.2 Error Distribution Characterization

The distribution of the output relative error can disclose useful information about the inferential multiplier behavior. In Figure 5.9 is reported the absolute value of the Relative Error (RE, normalized to the output dynamics) of both I-MULT and K-MULT computations. It appears clear that the statistical rules which I-MULT leverages to mimic the original logic function lead to more inaccurate computations w.r.t. K-MULT, both in average, 4.23% vs. 1.39%, and maximum value, 50.89% vs. 22.22%.

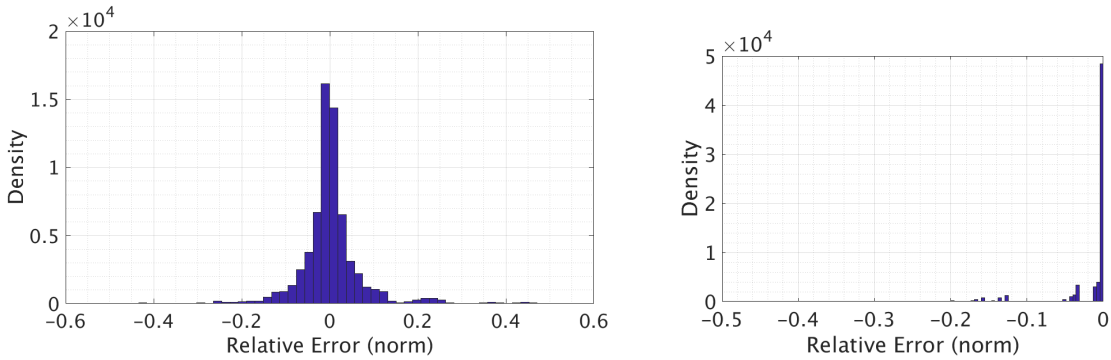


(a) I-MULT (b) K-MULT

Figure 5.9: Abs value of Relative Error Distribution for I-MULT (a) and K-MULT (b).

Although at first glance the error distribution of I-MULT could seem a severe impediment to its employment in error-resilient applications that requires a lower magnitude of computation error, the statistical nature of I-MULT errors reveals a possible strength point. Taking the relative error out of the absolute value, as reported in Figure 5.10, K-MULT error distribution results biased as the function under-design makes the approximate multiplications always smaller in value than the correct one. In contrast, the error distribution of I-MULT is almost specular around zero. This suggests that the accuracy drop due to the inference rules can be compensated if I-MULT is employed in Multiply-Accumulate (MAC) operations.

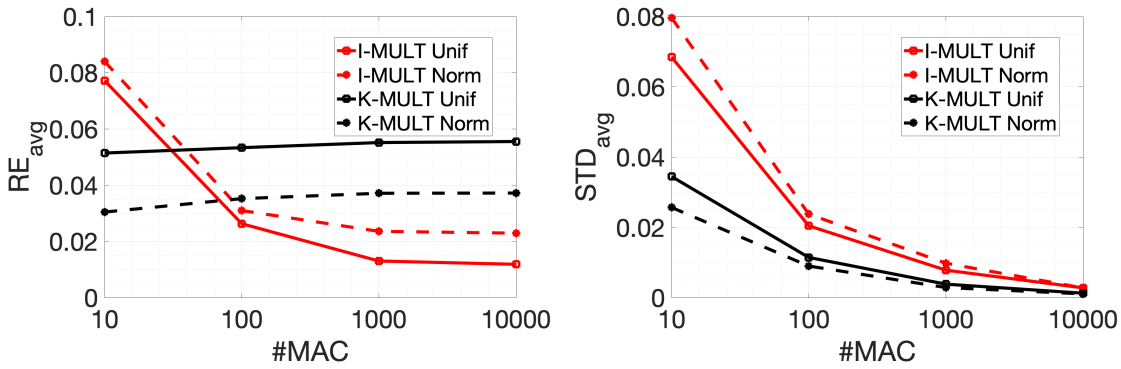
In order to prove that inferential multiplication enables statistical error compensation, I-MULT and K-MULT have been integrated into a MAC and simulated with two testbenches: *(i)* operands uniformly distributed in the range [0 - 255] and *(ii)* operands normally distributed [μ = 127, σ = 32]. We took 100 sets of [10, 100, 1000, 10000] MAC operations, and we collected the average RE and standard deviation (STD). The results



(a) I-MULT (b) K-MULT

Figure 5.10: Relative Error Distribution for I-MULT (a) and K-MULT (b).

depicted in Figure 5.11 show that for the I-MULT the higher the number of MAC operations the lower is the average RE. Also, starting from 100 MAC operations, I-MULT is always more accurate than K-MULT for both normal and uniform operand distribution. Nevertheless, K-MULT presents a lower standard deviation thanks to the arithmetic nature of its error. For the sake of clarity, it should be noticed that the statistical error compensation is strictly affected by the input distribution, thus, specific input patterns might be more favorable to K-MULT.



(a) Average Relative Error (b) Average Std Deviation

Figure 5.11: I-MULT and K-MULT average accuracy in MAC operations.

5.2.3 Final Considerations

The experimental results collected for the proposed multiplication by inference shows that such a method achieve reduced complexity, with area/latency large savings, at the cost of non-negligible accuracy loss. Although I-MULT does not represent a radical solution for efficient Energy-Accuracy scaling, it shows features that can be advantageous

in particular error-resilient applications. For example, as shown in the previous section, specific data-intensive application, e.g., Neural Networks, whose computation kernels are built upon MAC operations, can take advantages of I-MULT statistical error compensation. The demonstration of this hypothesis is out of the scope of this dissertation, but further exploration will be conducted in future work.

Chapter 6

Conclusions

The end of Moore's law is the prelude of a design crisis that will soon require to rethink the optimization and integration strategy of digital circuits and systems. Energy-Accuracy Scaling (EAS) does not represents a radical solution to all these concerns, however, is opening to new design paradigms that alleviate the pressure. This dissertation introduces new advanced design solutions that improve the approach the EAS paradigm is implemented. The presented two new strategies aim at reducing the design overhead of classical approximate solutions; according to the revisited taxonomy reported in this thesis, one of the proposed strategies belongs to the class of *Adaptive* EAS, while the second falls under the label of *Static* EAS.

The contributions of this dissertation on the *Adaptive* solutions concern the enhancement of the conventional Error Detection-Correction techniques for data-driven voltage scaling, i.e., Razor, in order to trade system accuracy for energy reduction. This mechanism, called *Approximate Error Detection-Correction* (AED-C), is built upon in-situ *elastic timing monitors* which enable an error management scheme suited to adaptive power management (e.g., Adaptive Voltage Over-Scaling) on error-resilient applications. AED-C implements EAS using the error detection coverage as a knob: a low error coverage encourage voltage over-scaling thus to achieve larger energy savings at the cost of quality-of-result; a high error coverage mitigate the voltage scaling leading to higher accuracy at the cost of lower energy savings. As EAS does not have to ensure full error coverage, the traditionally significant area/energy overhead of the conventional Razor may be reduced by using a simpler error management circuitry. Simulations over a representative set of applications/circuits, e.g., Multiply-Accumulate (MAC) unit, Discrete Cosine Transform (DCT), FIR and IIR filters, provide a comparative analysis with the state-of-the-art Razor. The collected results show that AED-C substantially reduces the average energy-per-operation (up to 44.7% savings w.r.t. Razor-driven Adaptive Voltage Over-Scaling) and the area overhead (3.3% vs. 62.0%), still guaranteeing reasonable accuracy. As a motivating example, when applied to a real-life image processing application, i.e., Discrete Cosine Transform Unit (DCT) integrated into a JPEG compressor, AED-C shows 51.9% energy savings (w.r.t. a baseline DCT implementation) and a

PSNR of 48.45 dB (w.r.t. baseline JPEG images)

Within the scope of *static* EAS, a new strategy is developed exploiting Machine Learning (ML) theories which suggest alternative forms to represent relationships among data. The proposed solution involves the design of Inferential Logic Circuits whose logic functions can be described as inference rules. Inspired by cognitive functions of the human brain, a machine learning-driven synthesis flows can map Boolean functions as Classification Trees that work like statistical inference engines. Circuits of this kind infer output values by evaluating the key features of the function learned during the training stage. The result is the design of combinational logic circuits that can mimic Boolean functions to a certain degree of accuracy. These inferential logic circuits run *quasi-exact* computations trading energy efficiency for accuracy in error-resilient applications. The Inferential Logic principle has been experimented on arithmetic circuits, more prone to support aggressive adaptive power management. The figures-of-merit of an *Inferential Multiplier* are quantified using representative image processing applications as a case study. A comparative analysis against a state-of-the-art Booth Multiplier proves the inferential logic representation simplifies the circuit complexity reducing the area by 22%. Also, the inferential multiplier can exploit 2× latency reduction for power optimization guaranteeing 76% average accuracy.

As a final remark, the research achievements of the proposed strategies are reported as follows:

- **Approximate Error Detection-Correction (AED-C) for Adaptive Energy-Accuracy Scaling (EAS)**:
  porting of the Approximate Computing concept to Adaptive EAS techniques using the timing faults coverage as a knob to trade energy for quality-of-results. AED-C offers across-level contributions in EDA methodology/tools, architectural solutions, and circuit implementation.

- **Inferential Logic for Static Energy-Accuracy Scaling**:
  porting of Machine Learning (ML) theories into the Boolean domain to implement logic functions described as statistical inference rules. The inferential logic circuits obtained through the proposed ML-based design flow present architectures prone to support aggressive Voltage (Frequency) scaling in exchange for a certain degree of accuracy.

Appendix A

List of Publications and Awards

International Journals

- **Roberto G. Rizzo**, Andea Calimera, and Jun Zhou., *Approximate Error Detection-Correction for efficient Adaptive Voltage Over-Scaling*. Integration, the VLSI Journal, Elsevier, Pages 220-231, 2018. DOI: 10.1016/j.vlsi.2018.04.008, ISSN: 0167-9260.
- **Roberto G. Rizzo**, Andea Calimera., *Implementing Adaptive Voltage Over-Scaling: Algorithm Noise Tolerance vs. Approximate Error Detection.* J. Low Power Electron. Appl. 2019, 9, 17.

Book Chapters

- Valentino Peluso, **Roberto G. Rizzo**, Andrea Calimera, Enrico Macii, and Massimo Alioto. *Beyond Ideal DVFS Through Ultra-Fine Grain Vdd-Hopping.* In VLSI-SoC: System-on-Chip in the Nanoscale Era – Design, Verification and Reliability, pages 152-172. Springer, 2017. DOI: 10.1007/978-3-319-67104-8-8, ISBN: 978-3-319-67103-1, 978-3-319-67104-8.
- **Roberto G. Rizzo**, Valentino Peluso, Andrea Calimera, and Jun Zhou. *On the Efficiency of Early Bird Sampling (EBS) An Error Detection-Correction Scheme for Data-Driven Voltage Overs-Scaling.* In VLSI-SoC: Opportunities and Challenges Beyond the Internet of Things, Springer, 2019

Proceedings of International Conferences

- **Roberto G. Rizzo**, Sandeep Miryala, Andrea Calimera, Enrico Macii, and Massimo Poncino. *Design and Characterization of Analog-to-Digital Converters using Graphene P-N Junctions.* In Proceedings of GLSVLSI 2015, pages 253-258, 2015. DOI: 10.1145/2742060.2742099, ISBN: 978-1-4503-3474-7.

- **Roberto G. Rizzo**, and Andrea Calimera. *Tunable Error Detection-Correction for Efficient Adaptive Voltage Over-Scaling*. In Proceedings of NGCAS 2017, pages 13-16, 2017. DOI: 10.1109/NGCAS.2017.75, ISBN: 978-1-5090-6447-2.

- **Roberto G. Rizzo**, Valentino Peluso, Andrea Calimera, Jun Zhou, and Xin Liu. *Early bird sampling: A short-paths free error detection-correction strategy for data-driven VOS.* In Proceedings of VLSI-SoC 2017, pages 1-6, 2017. DOI: 10.1109/VLSI-SoC.2017.8203472, ISBN: 978-1-5386-2880-5.

- **Roberto G. Rizzo**, Valerio Tenace, and Andrea Calimera. *Multiplication by Inference using Classification Trees: A Case-Study Analysis.* In Proceedings of ISCAS 2018, pages 1-5, 2018. DOI: 10.1109/ISCAS.2018.8351206, ISBN: 978-1-5386-4881-0.

- Valerio Tenace, **Roberto G. Rizzo**, Dejyoti Batthacaree, Andrea Calimera, and Anupam Chattopaday. *SAID: a Supergate-AIDed Logic Synthesis Flow for Memristive Crossbar.* In Proceedings of DATE 2019.

Awards

- **Golden Leaf Award** for the contribution *Tunable Error Detection-Correction for Efficient Adaptive VOS*, NGCAS 2017

Bibliography

[1] Massimo Alioto. "Energy-quality scalable adaptive VLSI circuits and systems beyond approximate computing". In: *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE. 2017, pp. 127–132.

[2] Massimo Alioto. "Ultra Low Power Design approaches for IoT". In: *Singapore-Hotchips* (2014).

[3] Massimo Alioto, Vivek De, and Andrea Marongiu. "Energy-quality scalable integrated circuits and systems: Continuing energy scaling in the twilight of Moore's law". In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 8.4 (2018), pp. 653–678.

[4] Haider AF Almurib, T Nandha Kumar, and Fabrizio Lombardi. "Inexact designs for approximate low power addition by cell replacement". In: *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2016, pp. 660–665.

[5] Todd Austin et al. "Opportunities and Challenges for Better Than Worst-case Design". In: *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*. ASP-DAC '05. Shanghai, China: ACM, 2005, pp. 2–7. ISBN: 0-7803-8737-6. DOI: 10.1145/1120725.1120878. URL: http://doi.acm.org/10.1145/1120725.1120878.

[6] Zdenka Babić, Aleksej Avramović, and Patricio Bulić. "An iterative logarithmic multiplier". In: *Microprocessors and Microsystems* 35.1 (2011), pp. 23–33.

[7] Pietro Babighian et al. "Post-layout Leakage Power Minimization Based on Distributed Sleep Transistor Insertion". In: *Proceedings of the 2004 International Symposium on Low Power Electronics and Design*. ISLPED '04. Newport Beach, California, USA: ACM, 2004, pp. 138–143. ISBN: 1-58113-929-2.

[8] R Bahar et al. "A symbolic method to reduce power consumption of circuits containing false paths". In: *Proceedings of the 1994 IEEE/ACM international conference on Computer-aided design*. IEEE Computer Society Press. 1994, pp. 368–371.

[9] Edith Beigné et al. "An asynchronous power aware and adaptive NoC based circuit". In: *IEEE Journal of Solid-State Circuits* 44.4 (Apr. 2009), pp. 1167–1177.

[10] Ahmed Benhassain et al. "Timing in-situ monitors: Implementation strategy and applications results". In: *2015 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE. 2015, pp. 1–4.

[11] L. Benini et al. "Battery-driven dynamic power management of portable systems". In: *Proceedings 13th International Symposium on System Synthesis*. 2000, pp. 25–30.

[12] Keith A Bowman et al. "Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance". In: *IEEE Journal of Solid-State Circuits* 44.1 (2009), pp. 49–63.

[13] Keith A Bowman et al. "A 45 nm resilient microprocessor core for dynamic variation tolerance". In: *IEEE Journal of Solid-State Circuits* 46.1 (2011), pp. 194–208.

[14] Keith A Bowman et al. "Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance". In: *IEEE Journal of Solid-State Circuits* 44.1 (2009), pp. 49–63.

[15] Leo Breiman et al. *Classification and regression trees*. CRC press, 1984.

[16] Melvin A Breuer. "Multi-media applications and imprecise computation". In: *8th Euromicro Conference on Digital System Design (DSD'05)*. IEEE. 2005, pp. 2–7.

[17] Henrik Brink, Joseph Richards, and Mark Fetherolf. *Real-world machine learning*. Manning Publications Co., 2016.

[18] A. Calimera et al. "Temperature-Insensitive Dual- V_{rmth} Synthesis for Nanometer CMOS Technologies Under Inverse Temperature Dependence". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 18.11 (Nov. 2010), pp. 1608–1620.

[19] Andrea Calimera et al. "Design of a family of sleep transistor cells for a clustered power-gating flow in 65nm technology". In: *Proceedings of the 17th ACM Great Lakes symposium on VLSI*. ACM. 2007, pp. 501–504.

[20] Marco Cannizzaro et al. "SafeRazor: Metastability-robust adaptive clocking in resilient circuits". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 62.9 (2015), pp. 2238–2247.

[21] Josep Carmona et al. "Elastic circuits". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28.10 (2009), pp. 1437–1455.

[22] L Ceze and J Larus. *Report on ISAT/DARPA Workshop on Accu- racy Trade-Offs Across the System Stack for Performance and Energy*. http://homes.cs.washington.edu/~luisceze/publications/ISAT_Accuracy_Trade-Offs_Across_the_System_Stack_for_Performance_and_Energy-PUBLIC-final.pptx. 2018.

[23] Ashutosh Chakraborty et al. “Dynamic thermal clock skew compensation using tunable delay buffers”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 16.6 (2008), pp. 639–649.

[24] Eugene Charniak. “The brain as a statistical inference engine—and you can too”. In: *Computational Linguistics* 37.4 (2011), pp. 643–655.

[25] Yu-Hsin Chen et al. “Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks”. In: *IEEE Journal of Solid-State Circuits* 52.1 (2017), pp. 127–138.

[26] Vinay K Chippa et al. “Analysis and characterization of inherent application resilience for approximate computing”. In: *Proceedings of the 50th Annual Design Automation Conference*. ACM. 2013, p. 113.

[27] Vinay K Chippa et al. “Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency”. In: *Proceedings of the 47th Design Automation Conference*. ACM. 2010, pp. 555–560.

[28] Mihir R Choudhury and Kartik Mohanram. “Approximate logic circuits for low overhead, non-intrusive concurrent error detection”. In: *Proceedings of the conference on Design, automation and test in Europe*. ACM. 2008, pp. 903–908.

[29] Shidhartha Das et al. “A 1 GHz hardware loop-accelerator with razor-based dynamic adaptation for energy-efficient operation”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 61.8 (2014), pp. 2290–2298.

[30] Shidhartha Das et al. “A self-tuning DVS processor using delay-error detection and correction”. In: *IEEE Journal of Solid-State Circuits* 41.4 (2006), pp. 792–804.

[31] Shidhartha Das et al. “RazorII: In situ error detection and correction for PVT and SER tolerance”. In: *IEEE Journal of Solid-State Circuits* 44.1 (2009), pp. 32–48.

[32] Sandeep Dhar, Dragan Maksimović, and Bruno Kranzen. “Closed-loop adaptive voltage scaling controller for standard-cell ASICs”. In: *Proceedings of the 2002 international symposium on Low power electronics and design*. ACM. 2002, pp. 103–107.

[33] Saurabh Dighe et al. “Within-Die Variation-Aware Dynamic-Voltage-Frequency-Scaling With Optimal Core Allocation and Thread Hopping for the 80-Core TeraFLOPS Processor”. In: *IEEE Journal of Solid-State Circuits* 46.1 (Nov. 2010), pp. 184–193.

[34] Monica Donno et al. “Enhanced Clustered Voltage Scaling for Low Power”. In: *Proceedings of the 12th ACM Great Lakes Symposium on VLSI*. GLSVLSI ’02. New York, New York, USA: ACM, 2002, pp. 18–23. ISBN: 1-58113-462-2. DOI: 10.1145/505306.505311. URL: http://doi.acm.org/10.1145/505306.505311.

[35] Ronald G Dreslinski et al. "Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits". In: *Proceedings of the IEEE* 98.2 (2010), pp. 253–266.

[36] J. Ebergen, J. Gainsley, and P. Cunningham. "Transistor sizing: how to control the speed and energy consumption of a circuit". In: *10th International Symposium on Asynchronous Circuits and Systems, 2004. Proceedings.* Apr. 2004, pp. 51–61. DOI: 10.1109/ASYNC.2004.1299287.

[37] Mohamed Elgebaly and Manoj Sachdev. "Variation-aware adaptive voltage scaling system". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 15.5 (2007), pp. 560–571.

[38] Dan Ernst et al. "Razor: A low-power pipeline based on circuit-level timing speculation". In: *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on.* IEEE. 2003, pp. 7–18.

[39] Dan Ernst et al. "Razor: circuit-level correction of timing errors for low-power operation". In: *IEEE Micro* 24.6 (2004), pp. 10–20.

[40] Krisztián Flautner and Trevor Mudge. "Vertigo: Automatic performance-setting for linux". In: *ACM SIGOPS Operating Systems Review* 36.SI (2002), pp. 105–116.

[41] Krisztián Flautner, Steve Reinhardt, and Trevor Mudge. "Automatic performance setting for dynamic voltage scaling". In: *Wireless networks* 8.5 (2002), pp. 507–520.

[42] Matthew Fojtik et al. "Bubble razor: Eliminating timing margins in an ARM cortex-M3 processor in 45 nm CMOS using architecturally independent error detection and correction". In: *IEEE Journal of Solid-State Circuits* 48.1 (2013), pp. 66–81.

[43] V Friedman and S Liu. "Dynamic logic CMOS circuits". In: *IEEE Journal of Solid-State Circuits* 19.2 (1984), pp. 263–266.

[44] Hiroshi Fuketa et al. "Adaptive performance compensation with in-situ timing error predictive sensors for subthreshold circuits". In: *IEEE Transactions on very large scale integration (VLSI) systems* 20.2 (2012), pp. 333–343.

[45] Samuel H Fuller and Lynette I Millett. *The Future of Computing Performance: Game Over or Next Level?* National Academy Press, 2011.

[46] Feng Gao and John P Hayes. "ILP-based optimization of sequential circuits for low power". In: *Proceedings of the 2003 international symposium on Low power electronics and design.* ACM. 2003, pp. 140–145.

[47] Joseph L Gastwirth. "The estimation of the Lorenz curve and Gini index". In: *The review of economics and statistics* (1972), pp. 306–316.

[48] Jason George et al. "Probabilistic arithmetic and energy efficient embedded signal processing". In: *Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems*. ACM. 2006, pp. 158–168.

[49] Kanad Ghose and Milind B. Kamble. "Reducing Power in Superscalar Processor Caches Using Subbanking, Multiple Line Buffers and Bit-line Segmentation". In: *Proceedings of the 1999 International Symposium on Low Power Electronics and Design*. ISLPED '99. San Diego, California, USA: ACM, 1999, pp. 70–75. ISBN: 1-58113-133-X. DOI: 10.1145/313817.313860. URL: http://doi.acm.org/10.1145/313817.313860.

[50] Swaroop Ghosh, Swarup Bhunia, and Kaushik Roy. "CRISTA: A new paradigm for low-power, variation-tolerant, and adaptive circuit synthesis using critical path isolation". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 26.11 (2007), pp. 1947–1956.

[51] Brian Greskamp et al. "Blueshift: Designing processors for timing speculation from the ground up." In: *2009 IEEE 15th International Symposium on High Performance Computer Architecture*. IEEE. 2009, pp. 213–224.

[52] Manuel de la Guia Solaz, Wei Han, and Richard Conway. "A flexible low power dsp with a programmable truncated multiplier". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 59.11 (2012), pp. 2555–2568.

[53] Onur Guzey et al. "Extracting a simplified view of design functionality based on vector simulation". In: *Haifa Verification Conference*. Springer. 2006, pp. 34–49.

[54] Abdul Hameed et al. "A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems". In: *Computing* 98.7 (July 2016), pp. 751–774. ISSN: 1436-5057. DOI: 10.1007/s00607-014-0407-8. URL: https://doi.org/10.1007/s00607-014-0407-8.

[55] Haibo He and Edwardo A Garcia. "Learning from imbalanced data". In: *IEEE Transactions on knowledge and data engineering* 21.9 (2009), pp. 1263–1284.

[56] Rajamohana Hegde and Naresh R Shanbhag. "A low-power digital filter IC via soft DSP". In: *Custom Integrated Circuits, 2001, IEEE Conference on*. IEEE. 2001, pp. 309–312.

[57] Rajamohana Hegde and Naresh R Shanbhag. "Energy-efficient signal processing via algorithmic noise-tolerance". In: *Low Power Electronics and Design, 1999. Proceedings. 1999 International Symposium on*. IEEE. 1999, pp. 30–35.

[58] Yen-Te Ho and Ting-Ting Hwang. "Low power design using dual threshold voltage". In: *Proceedings of the 2004 Asia and South Pacific Design Automation Conference*. IEEE Press. 2004, pp. 205–208.

[59] J. S. Hu et al. "Using dynamic branch behavior for power-efficient instruction fetch". In: *IEEE Computer Society Annual Symposium on VLSI, 2003. Proceedings*. Feb. 2003, pp. 127–132. DOI: 10.1109/ISVLSI.2003.1183363.

[60] *International Technology Roadmap for Semiconductors 2.0*. http://www.itrs2.net/. 2015.

[61] M. Jayakrishnan et al. "Slack-aware timing margin redistribution technique utilizing error avoidance flip-flops and time borrowing". In: *2015 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. Oct. 2015, pp. 159–164. DOI: 10.1109/VLSI-SoC.2015.7314409.

[62] Hailin Jiang, Malgorzata Marek-Sadowska, and Sani R Nassif. "Benefits and costs of power-gating technique". In: *2005 International Conference on Computer Design*. IEEE. 2005, pp. 559–566.

[63] Andrew B Kahng and Seokhyeong Kang. "Accuracy-configurable adder for approximate arithmetic designs". In: *Proceedings of the 49th Annual Design Automation Conference*. ACM. 2012, pp. 820–825.

[64] Andrew B Kahng et al. "Slack redistribution for graceful degradation under voltage overscaling". In: *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*. IEEE Press. 2010, pp. 825–831.

[65] Georgios Karakonstantis and Kaushik Roy. "Voltage over-scaling: A cross-layer design perspective for energy efficient systems". In: *Circuit Theory and Design (ECCTD), 2011 20th European Conference on*. IEEE. 2011, pp. 548–551.

[66] Lina Karam et al. "Trends in multicore DSP platforms". In: *IEEE signal processing magazine* 26.6 (2009), pp. 38–49.

[67] Jagrit Kathuria, M Ayoubkhan, and Arti Noor. "A review of clock gating techniques". In: *MIT International Journal of Electronics and Communication Engineering* 1.2 (2011), pp. 106–114.

[68] Himanshu Kaul et al. "A 1.45 GHz 52-to-162GFLOPS/W variable-precision floating-point fused multiply-add unit with certainty tracking in 32nm CMOS". In: *2012 IEEE International Solid-State Circuits Conference*. IEEE. 2012, pp. 182–184.

[69] Jongho Kim et al. "Delay monitoring system with multiple generic monitors for wide voltage range operation". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26.1 (2018), pp. 37–49.

[70] Seongjong Kim and Mingoo Seok. "Variation-tolerant, ultra-low-voltage microprocessor with a low-overhead, within-a-cycle in-situ timing-error detection and correction technique". In: *IEEE Journal of Solid-State Circuits* 50.6 (2015), pp. 1478–1490.

[71] Yong-Bin Kim. "Challenges for nanoscale MOSFETs and emerging nanoelectronics". In: *Transactions on Electrical and Electronic Materials* 11.3 (2010), pp. 93–105.

[72] Yongtae Kim, Yong Zhang, and Peng Li. "Energy efficient approximate arithmetic for error resilient neuromorphic computing". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23.11 (2015), pp. 2733–2737.

[73] H. Kojima, S. Tanaka, and K. Sasaki. "Half-swing clocking scheme for 75% power saving in clocking circuitry". In: *IEEE Journal of Solid-State Circuits* 30.4 (Apr. 1995), pp. 432–435. ISSN: 0018-9200. DOI: 10.1109/4.375963.

[74] Tejaswini Kolpe, Antonia Zhai, and Sachin S. Sapatnekar. "Enabling improved power management in multicore processors through clustered DVFS". In: *DATE'11: Design, Automation & Test in Europe Conference & Exhibition*. IEEE. Mar. 2011, pp. 1–6.

[75] Jonathan Koomey et al. "Implications of historical trends in the electrical efficiency of computing". In: *IEEE Annals of the History of Computing* 33.3 (2011), pp. 46–54.

[76] P. K. Krause and I. Polian. "Adaptive voltage over-scaling for resilient applications". In: *2011 Design, Automation Test in Europe*. Mar. 2011, pp. 1–6. DOI: 10.1109/DATE.2011.5763153.

[77] Parag Kulkarni, Puneet Gupta, and Milos Ercegovac. "Trading accuracy for power with an underdesigned multiplier architecture". In: *2011 24th Internatioal Conference on VLSI Design*. IEEE. 2011, pp. 346–351.

[78] Tadahiro Kuroda et al. "Variable supply-voltage scheme for low-power high-speed CMOS digital design". In: *IEEE Journal of Solid-State Circuits* 33.3 (1998), pp. 454–462.

[79] E. Kursun, S. Ghiasi, and M. Sarrafzadeh. "Transistor level budgeting for power optimization". In: *International Symposium on Signals, Circuits and Systems. Proceedings, SCS 2003. (Cat. No.03EX720)*. Mar. 2004, pp. 116–121. DOI: 10.1109/ISQED.2004.1283660.

[80] Inyong Kwon et al. "Razor-lite: a light-weight register for error detection by observing virtual supply rails". In: *IEEE Journal of Solid-State Circuits* 49.9 (2014), pp. 2054–2066.

[81] Khaing Yin Kyaw, Wang Ling Goh, and Kiat Seng Yeo. "Low-power high-speed multiplier for error-tolerant application". In: *2010 IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC)*. IEEE. 2010, pp. 1–4.

[82] Liangzhen Lai et al. "Slackprobe: A low overhead in situ on-line timing slack monitoring methodology". In: *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium. 2013, pp. 282–287.

[83] Wan-Ping Lee, Hung-Yi Liu, and Yao-Wen Chang. "Voltage island aware floorplanning for power and timing optimization". In: *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*. ACM. 2006, pp. 389–394.

[84] Larkhoon Leem et al. "ERSA: Error resilient system architecture for probabilistic applications". In: *Proceedings of the Conference on Design, Automation and Test in Europe*. European Design and Automation Association. 2010, pp. 1560–1565.

[85] Xuanhua Li and Donald Yeung. "Application-level correctness and its impact on fault tolerance". In: *2007 IEEE 13th International symposium on high performance computer architecture*. IEEE. 2007, pp. 181–192.

[86] Avinash Lingamneni et al. "Energy parsimonious circuit design through probabilistic pruning". In: *2011 Design, Automation & Test in Europe*. IEEE. 2011, pp. 1–6.

[87] Bin Liu et al. "Power driven placement with layout aware supply voltage assignment for voltage island generation in Dual-Vdd designs". In: *Asia and South Pacific Conference on Design Automation, 2006.* IEEE. 2006, 6–pp.

[88] Cong Liu, Jie Han, and Fabrizio Lombardi. "A low-power, high-performance approximate multiplier with configurable partial error recovery". In: *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2014, pp. 1–4.

[89] Jane WS Liu et al. "Imprecise computations". In: *Proceedings of the IEEE* 82.1 (1994), pp. 83–94.

[90] X. Liu et al. "An Ultralow-Voltage Sensor Node Processor With Diverse Hardware Acceleration and Cognitive Sampling for Intelligent Sensing". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 62.12 (Dec. 2015), pp. 1149–1153. ISSN: 1549-7747. DOI: 10.1109/TCSII.2015.2468927.

[91] Yang Liu and Tong Zhang. "On the selection of arithmetic unit structure in voltage overscaled soft digital signal processing". In: *Proceedings of the 2007 international symposium on Low power electronics and design*. ACM. 2007, pp. 250–255.

[92] Shih-Lien Lu. "Speeding up processing with approximation circuits". In: *Computer* 37.3 (2004), pp. 67–73.

[93] Jeffrey T Ludwig, S Hamid Nawab, and Anantha P Chandrakasan. "Low-power digital filtering using approximate processing". In: *IEEE Journal of Solid-State Circuits* 31.3 (1996), pp. 395–400.

[94] Hamid Reza Mahdiani et al. "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 57.4 (2010), pp. 850–862.

[95] Rahul Mangharam and Aminreza Abrahimi Saba. "Anytime algorithms for GPU architectures". In: *2011 IEEE 32nd Real-Time Systems Symposium*. IEEE. 2011, pp. 47–56.

[96] Y. Masuda et al. "Comparing Voltage Adaptation Performance between Replica and In-Situ Timing Monitors". In: *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. Nov. 2018, pp. 1–8. DOI: 10.1145/3240765.3240788.

[97] Jin Miao et al. "Modeling and synthesis of quality-energy optimal approximate adders". In: *Proceedings of the International Conference on Computer-Aided Design*. ACM. 2012, pp. 728–735.

[98] Sylvain Miermont, Pascal Vivet, and Marc Renaudin. "A power supply selector for energy-and area-efficient local dynamic voltage scaling". In: *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*. Vol. 4644. Springer, 2007, pp. 556–565.

[99] Debabrata Mohapatra, Georgios Karakonstantis, and Kaushik Roy. "Low-power process-variation tolerant arithmetic units using input-based elastic clocking". In: *Proceedings of the 2007 international symposium on Low power electronics and design*. ACM. 2007, pp. 74–79.

[100] Debabrata Mohapatra et al. "Design of voltage-scalable meta-functions for approximate computing". In: *2011 Design, Automation & Test in Europe*. IEEE. 2011, pp. 1–6.

[101] José Monteiro and Srinivas Devadas. "Techniques for power estimation and optimization at the logic level: a survey". In: *Journal of VLSI signal processing systems for signal, image and video technology* 13.2-3 (1996), pp. 259–276.

[102] Bert Moons and Marian Verhelst. "Dvas: Dynamic voltage accuracy scaling for increased energy-efficiency in approximate computing". In: *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE. 2015, pp. 237–242.

[103] Bert Moons et al. "14.5 envision: A 0.26-to-10tops/w subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm fdsoi". In: *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE. 2017, pp. 246–247.

[104] Bert Moons et al. "DVAFS: Trading computational accuracy for energy through dynamic-voltage-accuracy-frequency-scaling". In: *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE. 2017, pp. 488–493.

[105] Gordon E Moore et al. *Cramming more components onto integrated circuits*. 1965.

[106] Srinivasan Narayanamoorthy et al. "Energy-efficient approximate multiplication for digital signal processing and classification applications". In: *IEEE transactions on very large scale integration (VLSI) systems* 23.6 (2015), pp. 1180–1184.

[107] Mehrzad Nejat, Bijan Alizadeh, and Ali Afzali-Kusha. "Dynamic flip-flop conversion to tolerate process variation in low power circuits". In: *Proceedings of the conference on Design, Automation & Test in Europe*. European Design and Automation Association. 2014, p. 111.

[108] David Nguyen et al. "Minimization of dynamic and static power through joint assignment of threshold voltages and sizing optimization". In: *Proceedings of the 2003 international symposium on Low power electronics and design*. ACM. 2003, pp. 158–163.

[109] Erwan Nogues, Daniel Menard, and Maxime Pelcat. "Algorithmic-level approximate computing applied to energy efficient hevc decoding". In: *IEEE Transactions on Emerging Topics in Computing* (2016).

[110] K. J. Nowka et al. "A 32-bit PowerPC system-on-a-chip with support for dynamic voltage scaling and dynamic frequency scaling". In: *IEEE Journal of Solid-State Circuits* 37.11 (Nov. 2002), pp. 1441–1447.

[111] Damian Nowroth, Ilia Polian, and Bernd Becker. "A study of cognitive resilience in a JPEG compressor". In: *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*. IEEE. 2008, pp. 32–41.

[112] Daniele Jahier Pagliari et al. "An automated design flow for approximate circuits based on reduced precision redundancy". In: *Computer Design (ICCD), 2015 33rd IEEE International Conference on*. IEEE. 2015, pp. 86–93.

[113] Krishna V Palem et al. "Sustaining moore's law in embedded computing through probabilistic and approximate design: retrospects and prospects". In: *Proceedings of the 2009 international conference on Compilers, architecture, and synthesis for embedded systems*. ACM. 2009, pp. 1–10.

[114] Veera Papirla, Aarul Jain, and Chaitali Chakrabarti. "Low power robust signal processing". In: *Proceedings of the 2009 ACM/IEEE international symposium on Low power electronics and design*. ACM. 2009, pp. 303–306.

[115] J. Park et al. "Accurate modeling and calculation of delay and energy overheads of dynamic voltage scaling in modern high-performance microprocessors". In: *2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*. Aug. 2010, pp. 419–424.

[116] Jongsun Park, Jung Hwan Choi, and Kaushik Roy. "Dynamic bit-width adaptation in DCT: An approach to trade off image quality and computation energy". In: *IEEE transactions on very large scale integration (VLSI) systems* 18.5 (2010), pp. 787–793.

[117] Jeff Parkhurst, John Darringer, and Bill Grundmann. "From single core to multi-core: preparing for a new exponential". In: *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*. ACM. 2006, pp. 67–72.

[118] Valentino Peluso et al. “Beyond Ideal DVFS Through Ultra-Fine Grain Vdd-Hopping”. In: *IFIP/IEEE International Conference on Very Large Scale Integration-System on a Chip*. Springer. 2016, pp. 152–172.

[119] Valentino Peluso et al. “Ultra-Fine Grain Vdd-Hopping for energy-efficient Multi-Processor SoCs”. In: *2016 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. IEEE. Sept. 2016, pp. 1–6.

[120] T. Pering, T. Burd, and R. Brodersen. “The simulation and evaluation of dynamic voltage scaling algorithms”. In: *Proceedings. 1998 International Symposium on Low Power Electronics and Design (IEEE Cat. No.98TH8379)*. Aug. 1998, pp. 76–81. DOI: 10.1145/280756.280790.

[121] J.M. Rabaey and M. Pedram. *Low Power Design Methodologies*. The Springer International Series in Engineering and Computer Science. Springer US, 2012. ISBN: 9781461523079. URL: https://books.google.it/books?id=9IzuBwAAQBAJ.

[122] Jan M. Rabaey. *Digital Integrated Circuits: A Design Perspective*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996. ISBN: 0-13-178609-1.

[123] Abbas Rahimi, Luca Benini, and Rajesh K Gupta. “Variability mitigation in nanometer CMOS integrated systems: A survey of techniques from circuits to software”. In: *Proceedings of the IEEE* 104.7 (2016), pp. 1410–1448.

[124] Ashish Ranjan et al. “ASLAN: Synthesis of approximate sequential circuits”. In: *Proceedings of the conference on Design, Automation & Test in Europe*. European Design and Automation Association. 2014, p. 364.

[125] Roberto G. Rizzo and Andrea Calimera. “Tunable Error Detection-Correction for Efficient Voltage Over-Scaling”. In: *New Generation of Circuits and Systems Conference (NGCAS), 2017 IEEE 1st International*. IEEE. 2017.

[126] Roberto G. Rizzo et al. “Early Bird Sampling: a Short-Paths Free Error Detection-Correction Strategy for Data-Driven VOS”. In: *International Conference on Very Large Scale Integration (VLSI-SoC)), 2017 IEEE 25th International*. IEEE. 2017.

[127] Roberto G Rizzo, Andrea Calimera, and Jun Zhou. “Approximate Error Detection-Correction for efficient Adaptive Voltage Over-Scaling”. In: *Integration* 63 (2018), pp. 220–231.

[128] Roberto Giorgio Rizzo and Andrea Calimera. “Implementing Adaptive Voltage Over-Scaling: Algorithmic Noise Tolerance vs. Approximate Error Detection”. In: *Journal of Low Power Electronics and Applications* 9.2 (2019). ISSN: 2079-9268. DOI: 10.3390/jlpea9020017. URL: http://www.mdpi.com/2079-9268/9/2/17.

[129] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand. "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits". In: *Proceedings of the IEEE* 91.2 (Feb. 2003), pp. 305–327. ISSN: 0018-9219. DOI: 10.1109/JPROC.2002.808156.

[130] M. Saliva et al. "Digital circuits reliability with in-situ monitors in 28nm fully depleted SOI". In: *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*. Mar. 2015, pp. 441–446. DOI: 10.7873/DATE.2015.0238.

[131] T. Sato and Y. Kunitake. "A Simple Flip-Flop Circuit for Typical-Case Designs for DFM". In: *8th International Symposium on Quality Electronic Design (ISQED'07)*. Mar. 2007, pp. 539–544. DOI: 10.1109/ISQED.2007.23.

[132] Michael J Schulte, James E Stine, and John G Jansen. "Reduced power dissipation through truncated multiplication". In: *Proceedings IEEE Alessandro Volta Memorial Workshop on Low-Power Design*. IEEE. 1999, pp. 61–69.

[133] Michael J Schulte and Earl E Swartzlander. "A family of variable-precision interval arithmetic processors". In: *IEEE Transactions on Computers* 49.5 (2000), pp. 387–397.

[134] W. Shan, L. Shi, and J. Yang. "In-SituTiming Monitor-Based Adaptive Voltage Scaling System for Wide-Voltage-Range Applications". In: *IEEE Access* 5 (2017), pp. 15831–15838. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2670644.

[135] Youhua Shi et al. "Suspicious timing error prediction with in-cycle clock gating". In: *International Symposium on Quality Electronic Design (ISQED)*. IEEE. 2013, pp. 335–340.

[136] Byonghyo Shim and Naresh R Shanbhag. "Reduced precision redundancy for low-power digital filtering". In: *Conference Record of Thirty-Fifth Asilomar Conference on Signals, Systems and Computers (Cat. No. 01CH37256)*. Vol. 1. IEEE. 2001, pp. 148–152.

[137] Byonghyo Shim, Srinivasa R Sridhara, and Naresh R Shanbhag. "Reliable low-power digital signal processing via reduced precision redundancy". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 12.5 (2004), pp. 497–510.

[138] Doochul Shin and Sandeep K Gupta. "Approximate logic synthesis for error tolerant applications". In: *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*. IEEE. 2010, pp. 957–960.

[139] Stelios Sidiroglou-Douskos et al. "Managing performance vs. accuracy trade-offs with loop perforation". In: *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*. ACM. 2011, pp. 124–134.

[140] Tom Simonite. "Moore's Law is dead. Now what". In: *MIT Technology Review* (2016).

[141] Amit Sinha and Anantha P Chandrakasan. "Energy efficient filtering using adaptive precision and variable voltage". In: *Twelfth Annual IEEE International ASIC/SOC Conference (Cat. No. 99TH8454)*. IEEE. 1999, pp. 327–331.

[142] S. M. Srinivas Devadas. "A Survey of Optimization Techniques Targeting Low Power VLSI Circuits". In: *32nd Design Automation Conference*. June 1995, pp. 242–247. DOI: 10.1109/DAC.1995.250098.

[143] Anup Kumar Sultania, Dennis Sylvester, and Sachin S Sapatnekar. "Transistor and pin reordering for gate oxide leakage reduction in dual T/sub ox/circuits". In: *IEEE International Conference on Computer Design: VLSI in Computers and Processors, 2004. ICCD 2004. Proceedings.* IEEE. 2004, pp. 228–233.

[144] Chin Hwee Tan. "Minimization of power in VLSI circuits using transistor sizing, input ordering, and statistical power estimation". In: *Proc. IWLPD*. 1994, pp. 75–80.

[145] Valerio Tenace and Andrea Calimera. "Activation-Kernel Extraction through Machine Learning". In: *CAS (NGCAS), 2017 New Generation of*. IEEE. 2017, pp. 5–8.

[146] Valerio Tenace and Andrea Calimera. "Inferential Logic: a Machine Learning Inspired Paradigm for Combinational Circuits". In: *2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. IEEE. 2018, pp. 149–154.

[147] Valerio Tenace and Andrea Calimera. "Quasi-exact logic functions through classification trees". In: *Integration* 63 (2018), pp. 248–255.

[148] Joshua B Tenenbaum et al. "How to grow a mind: Statistics, structure, and abstraction". In: *science* 331.6022 (2011), pp. 1279–1285.

[149] Dean N. Truong et al. "A 167-processor computational platform in 65 nm CMOS". In: *IEEE Journal of Solid-State Circuits* 44.4 (Apr. 2009), pp. 1130–1144.

[150] James W Tschanz et al. "Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage". In: *IEEE Journal of Solid-State Circuits* 37.11 (2002), pp. 1396–1402.

[151] James Tschanz et al. "Tunable replica circuits and adaptive voltage-frequency techniques for dynamic voltage, temperature, and aging variation tolerance". In: *2009 Symposium on VLSI Circuits*. IEEE. 2009, pp. 112–113.

[152] Kimiyoshi Usami and Mark Horowitz. "Clustered Voltage Scaling Technique for Low-power Design". In: *Proceedings of the 1995 International Symposium on Low Power Design*. ISLPED '95. Dana Point, California, USA: ACM, 1995, pp. 3–8. ISBN: 0-89791-744-8. DOI: 10.1145/224081.224083. URL: http://doi.acm.org/10.1145/224081.224083.

[153] Kimiyoshi Usami and Mark Horowitz. "Clustered voltage scaling technique for low-power design". In: *ISLPD*. Vol. 95. 1995, pp. 3–8.

[154] Stefanos Valadimas, Yiorgos Tsiatouhas, and Angela Arapoyanni. "Timing error tolerance in nanometer ICs". In: *On-Line Testing Symposium (IOLTS), 2010 IEEE 16th International*. IEEE. 2010, pp. 283–288.

[155] Rakesh Vattikonda, Wenping Wang, and Yu Cao. "Modeling and minimization of PMOS NBTI effect for robust nanometer design". In: *Proceedings of the 43rd annual Design Automation Conference*. ACM. 2006, pp. 1047–1052.

[156] Vasanth Venkatachalam and Michael Franz. "Power Reduction Techniques for Microprocessor Systems". In: *ACM Comput. Surv.* 37.3 (Sept. 2005), pp. 195–237. ISSN: 0360-0300. DOI: 10.1145/1108956.1108957. URL: http://doi.acm.org/10.1145/1108956.1108957.

[157] Vasanth Venkatachalam and Michael Franz. "Power reduction techniques for microprocessor systems". In: *ACM Computing Surveys (CSUR)* 37.3 (Sept. 2005), pp. 195–237.

[158] Swagath Venkataramani et al. "Quality programmable vector processors for approximate computing". In: *2013 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE. 2013, pp. 1–12.

[159] Swagath Venkataramani et al. "SALSA: systematic logic synthesis of approximate circuits". In: *DAC Design Automation Conference 2012*. IEEE. 2012, pp. 796–801.

[160] Li-C Wang. "Experience of Data Analytics in EDA and Test—Principles, Promises, and Challenges". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 36.6 (2017), pp. 885–898.

[161] Yu-Ming Yang, Iris Hui-Ru Jiang, and Sung-Ting Ho. "PushPull: Short-path padding for timing error resilient circuits". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33.4 (2014), pp. 558–570.

[162] Kazuo Yano et al. "Top-down pass-transistor logic design". In: *IEEE journal of solid-state circuits* 31.6 (1996), pp. 792–803.

[163] Muhammad Zakarya and Lee Gillam. "Energy efficient computing, clusters, grids and clouds: A taxonomy and survey". In: *Sustainable Computing: Informatics and Systems* 14 (2017), pp. 13–33. ISSN: 2210-5379. DOI: https://doi.org/10.1016/j.suscom.2017.03.002. URL: http://www.sciencedirect.com/science/article/pii/S2210537917300707.

[164] Bo Zhai et al. "Theoretical and practical limits of dynamic voltage scaling". In: *Proceedings of the 41st annual Design Automation Conference*. ACM. 2004, pp. 868–873.

[165] Jun Zhou et al. "HEPP: A new in-situ timing-error prediction and prevention technique for variation-tolerant ultra-low-voltage designs". In: *Solid-State Circuits Conference (A-SSCC), 2013 IEEE Asian*. IEEE. 2013, pp. 129–132.

[166] Ning Zhu et al. “Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 18.8 (2010), pp. 1225–1229.

[167] Ning Zhu et al. “Enhanced low-power high-speed adder for error-tolerant application”. In: *2010 International SoC Design Conference*. IEEE. 2010, pp. 323–327.